Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Vanja Mileski

Evalvacija algoritmov za diarizacijo govorcev v zvočnih posnetkih

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana, 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu se posvetite primerjavi pristopov za avtomatično diarizacijo govorcev v zvočnih posnetkih. Preučite področje diarizacije in pridobite čimveč različnih algoritmov za diarizacijo. Izberite nabor zvočnih posnetkov in jih ročno anotirajte, nato pa ocenite pridobljene algoritme in preučite kateri so najbolj primerni za različne vrste posnetkov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Vanja Mileski z vpisno številko **63120316** sem avtor diplomskega dela z naslovom:

Evalvacija algoritmov za diarizacijo govorcev v zvočnih posnetkih

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 1.09.2015 Podpis avtorja:

Zahvaljujem se mentorju, doc. dr. Matiji Maroltu, za predloge in pomoč pri izdelavi diplomskega dela. Zahvaljujem se družini in vsem prijateljem za motivacijo v času študija.

Družini in prijateljem

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Diarizacija **5**

2.1 Ekstrakcija značilk 5

2.1.1 Linearno napovedno kodiranje (LPC) 6

2.1.2 LPC izpeljani kepstralni koeficienti 7

2.1.3 Linearni spektralni pari (LSP) 7

2.1.4 Mel-frekvenčni kepstralni koeficienti (MFCC) 7

2.1.5 Kratkoročna energija (STE) 8

2.1.6 Višina tona 8

2.2 Segmentacija 9

2.2.1 Segmentacija na osnovi modela 9

2.2.1.1 Prikriti Markovski modeli (HMM) 10

2.2.1.2 Mešanice Gaussovih porazdelitev (GMM) . . 11

2.2.1.3 Viterbijev algoritem 11

2.2.2 Segmentacija na osnovi metrike 13

2.2.2.1 Bayesov informacijski kriterij (BIC) 13

2.2.3 Hibridna segmentacija 14

2.2.4 Vrednotenje uspešnosti segmentacije 14

2.3 Gručenje 16

2.3.1 Deterministične metode 16
2.3.1.1 Metode, ki temeljijo na SOM 16
2.3.1.2 Hierarhične metode 16
2.3.2 Verjetnostne metode 18
2.3.2.1 Metode, ki temeljijo na GMM 18
2.3.2.2 Metode, ki temeljijo na HMM 19

3 Orodja za diarizacijo govorcev 21
3.1 LIUM Speaker diarization 21
3.2 PyAudioAnalysis 24
3.3 Diarize-jruby 26
3.4 VoiceID 27
3.5 ALIZE 29
3.6 Matlab Audio Analysis Library 31

4 Evalvacija in rezultati 35
4.1 Ročna diarizacija testnih posnetkov 35
4.2 Vizualizacija ročne diarizacije 37
4.3 Ocenjevanje algoritmov 40
4.4 Rezultati 41
4.4.1 LIUM - rezultati 41
4.4.2 PyAA - rezultati 43
4.4.3 Diarize-jruby - rezultati 45
4.4.4 VoiceID - rezultati 47
4.4.5 Alize - rezultati 49
4.4.6 MAAL - rezultati 50
4.5 Primerjava algoritmov in analiza rezultatov 52

5 Sklepne ugotovitve 55

Literatura 57

Seznam uporabljenih kratic

kratica	angleško	slovensko
WAV	Waveform Audio format	Zvočni format, ki uporablja valovne krivulje
MFCC	Mel-Frequency Cepstral Coefficient	Mel-frekvenčni kepstralen koeficient
LPC	Linear Prediction Coefficients	Linearno napovedno kodiranje
PLP	Perceptual Linear Predictive	Zaznavno linearno napovedno kodiranje
LSP	Line Spectral Pairs	Linijski spektralni pari
ZCR	Zero-Crossing Rate	Stopnja ničelnega prehoda
STE	Short-Time Energy	Kratkoročna energija
DFT	Discrete Fourier Transform	Diskretna Fourierjeva transformacija
FFT	Fast Fourier Transform	Hitra Fourierjeva transformacija
UBM	Universal Background Model	Univerzalni model ozadja
SSM	Sample Speaker Model	Model vzorčnih govorcev
HMM	Hidden Markov Model	Prikriti Makrovski modeli
SVM	Support Vector Machines	Metoda podpornih vektorjev
PDF	Probability Distribution Function	Funkcija verjetnostne porazdelitve
GMM	Gaussian Mixture Models	Mešanice Gaussovih porazdelitev
EM	Expectation Maximization	Maksimizacija pričakovanja
GLR	Generalized Likelihood Ratio	Generalizirano razmerje verjetnosti

BIC	Bayes Information Criterion	Bayesov informacijski kriterij
FAR	False Alarm Rate	Stopnja napačnih pozitivnih primerov
MDR	Miss Detection Rate	Stopnja neuspešnega odkrivanja
SER	Speaker Error Rate	Stopnja napake govorcev
SOM	Self-Organizing Map	Samoorganizirajoča mreža
CSM	Cosine Similarity Measure	Kosinusna podobnost
DER	Diarization Error Rate	Stopnja napak diarizacije
JAR	Java Archive	Javanska knjižnica
NIST	National Institute of Standards and Technology	Nacionalni inštitut za standarde in tehnologijo (ZDA)

Povzetek

Segmentacija govorcev (diarizacija) je postopek, ki razdeli zvočni posnetek na odseke glede na identiteto govorcev. Segmentacija govorcev v zvočnih posnetkih nam da odgovor na vprašanje *kdo je kdaj govoril.*

V tem diplomskem delu se posvetimo avtomatični segmentaciji govorcev v različnih zvočnih posnetkih. Pripravimo si testno množico zvočnih posnetkov v slovenščini, ki jih pridobimo iz terenskih posnetkov. Posnetki vsebujejo dva ali več govorcev in zelo pogosto tudi druge zvoke, tišino, prekrivanje med govorcema in podobno. Zanje ročno zgradimo transkripcije za število govorcev in časovni interval govora posameznega govorca, ki bo predstavljal našo resnico (angl. *ground-truth*). Nad testno množico poženemo vse algoritme za diarizacijo, ki jih ocenjujemo. Napišemo program, ki za vhod vzame rezultate vseh algoritmov, ki imajo različne formate in vrste zastopanja rezultatov, ter jih pretvorimo v enotno obliko. Ocenjujemo natančnost algoritmov in analiziramo, kako dobro delajo v različnih situacijah.

Ključne besede: segmentacija govorcev, diarizacija, terenski posnetki, govor, evalvacija algoritmov.

Abstract

Speaker segmentation (diarisation, diarization) is a process that separates the audio clip in sections regarding the identity of the speakers. Speaker diarization in sound recordings answers the question of *who spoke when?*

This thesis is dedicated to the automatic segmentation of speakers in a variety of sound recordings. We prepare a test data of audio recordings in Slovenian, which are obtained from field recordings. The recordings contain two or more speakers and very often they contain other sounds, silence, overlap between the speakers and the like. We manually build transcriptions for the number of speakers in them and the time interval of the speeches for each speaker which will represent our ground-truth. We run all the algorithms for diarization that we evaluate on this test data. We write a program which takes the results from the algorithms as an input which have different formats and different types of representation, and the results are converted to a common format. We evaluate the accuracy of the algorithms and analyse how well they work in different situations.

Keywords: speaker segmentation, diarization, diarisation, field recordings, speech, algorithm evaluation.

Poglavje 1

Uvod

V vsakdanjem življenju naletimo na številne tehnologije, ki uporabljajo zvok kot glavni vir za izmenjavo informacij, od filmov, televizijskih serij, novic, televizijskih in radio oddaj do posnetkov s poslovnih sestankov, glasovnih sporočil, glasovne pošte in tako naprej. Zaznavanje, prepoznavanje in diarizacija govora so postali zelo zanimiv predmet razprav in raziskav. V tem diplomskem delu bomo pogledali le diarizacijo govorcev (ang. *diarisation* ali *diarization*).

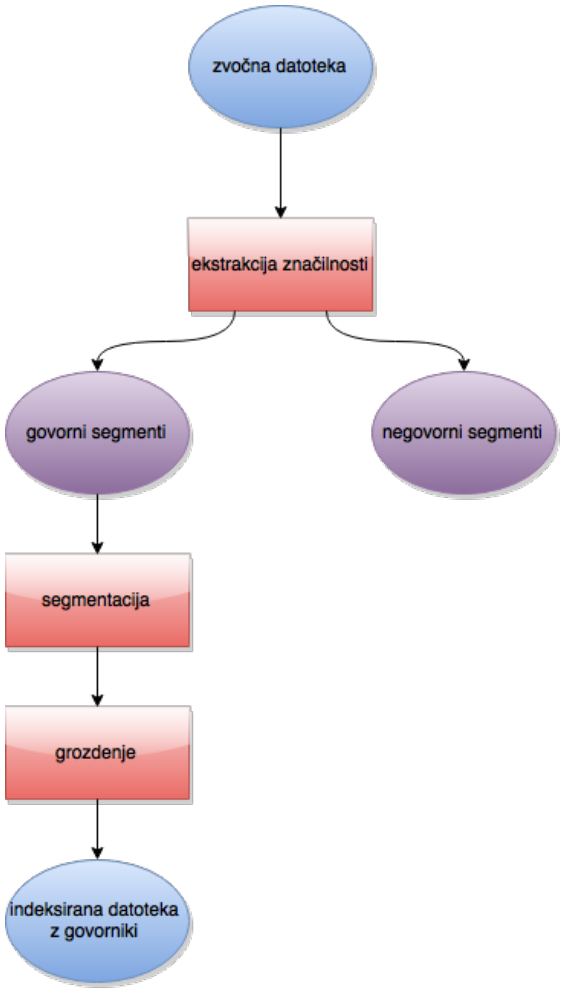
Diarizacija je proces avtomatične razdelitve zvočnega posnetka na segmente govorcev in ugotavljanje, kateri segmenti so od istega govorca. Uporablja se za odgovor na vprašanje, *"kdo je kdaj govoril,"* in lahko izboljša berljivost avtomatske transkripcije govora s strukturiranjem zvočnega toka v niz govorcev [1] [2]. Prvi korak je ekstrakcija značilk zvočnega posnetka, s katerimi parametri se lahko začne diarizacija (slika 1.1). Proces diarizacije je sestavljen iz dveh delov, to sta segmentacija in gručenje (ang. *clustering*) govorcev. Prvi je namenjen iskanju točk, pri čemer pride do sprememb govorcev v zvočnem posnetku - razdeli zvok v akustično homogene segmente, tako da vsak segment vsebuje (v najboljšem primeru) samo enega govorca. Cilj drugega je nenadzorovana uvrstitev govornih segmentov in njihovo združevanje na podlagi značilke govorca. To pomeni, da prepoznajo vse govorne segmente, ki jih izgovori isti govorec v zvočnem posnetku, jih združi

in jim dodeliti skupno oznako. Ta proces se lahko naredi od spodaj navzgor (ang. *bottom-up*) ali od zgoraj navzdol (ang. *top-down*)[1]. Čeprav sta ta dva procesa lahko skupaj optimizirana ali narejena v obratnem vrstnem redu, je običajno prvi korak segmentacija, čemur sledi gručenje govorcev.

Diarizacija, če pogledamo hitrost sprememb tehnologije v današnjem času, ni relativno nova stvar. Obstaja že veliko število pristopov in algoritmov na tem področju; Nacionalni inštitut za standarde in tehnologijo[1] (ZDA) ima projekt za vrednotenje bogate transkripcije (ang. *Rich Transcription Evaluation*) [2] [1] [3], vendar poleg tega, da se ta tehnologija že dolgo čas razvija, še vedno ni tako dobra, kot si želimo. Največ težav se pojavlja skoraj pri vseh algoritmih takrat, ko posnetki vsebujejo šum, hrup, prekrivanja med govorci in razlike v glasovni moči govorcev. V tem diplomskem delu bomo poleg teh "težav"delali še strožje ocenjevanje, ker bomo evalvacijo delali na posnetkih, ki so izključno v slovenskem jeziku. To je tudi na neki način motivacija za to diplomsko nalogo, saj so (skoraj) vsi obstoječi algoritmi izdelani in preizkušeni samo na angleških govorih. Največji cilj tega dela bo dosežen, če spodbudi izdelavo algoritma (ali optimizacijo že obstoječih) za slovenske ali večjezične posnetke.

Delo je razdeljeno na več poglavij. V naslednjem poglavju bomo pregledali področje, ki ga obsega ta diplomska naloga, in predstavili ozadje problema z opisom nekatere najpogosteje uporabljenih algoritmov. V tretjem poglavju bomo bolj natančno pogledali različne algoritme za diarizacijo govorcev, ki so eni izmed najbolj popularnih algoritmov v tem trenutku. Četrto poglavje opisuje evalvacijo, ki je bila narejena z lastnim programom, in poda ter analizira dobljene rezultate. Peto poglavje sestavljajo sklepne ugotovitve.

[1](ang. *National Institute of Standards and Technology - NIST*)



Slika 1.1: Tipičen postopek diarizacije

Poglavje 2

Diarizacija

2.1 Ekstrakcija značilk

Prvi korak pri pridobivanju sekvence akustičnih opazovanj je pretvorba analognega zvočnega signala v digitalno predstavitev. Pri analogno-digitalni pretvorbi je amplituda signala izmerjena v določenih časovnih intervalih in zapisana kot število s plavajočo vejico. Ker je v tem zaporedju številk veliko podatkov, ki so odveč, se preoblikujejo tako, da se ohranijo le pomembne informacije, vendar so zdaj podatki manj redundantni [2]. Ta korak se imenuje ekstrakcija značilk. Najprej se naredi spektralna analiza na zaporedju amplitude. Ti spektralni podatki se nato uporabijo kot vhod za filter, ki spreminja podatke po modelu, primernem za človeški sluh. Dve pogosto uporabljeni metodi za ta postopek sta: analiza z uporabo mel-frekvenčnega kepstralnega koeficienta (ang. *Mel-Frequency Cepstral Coefficient*) in analiza z uporabo zaznavnega linearnega napovednega kodiranja (ang. *Perceptual Linear Predictive*). Obe metodi dasta na izhod serijo vektorjev. V zadnjem koraku se prvi in drugi odvodi pogosto združijo z verigo teh vektorjev lastnosti.

$$x[n; m], n = m - N_{sl} + 1, m - N_{sl} + 2, ..., m \tag{2.1}$$

Naj (2.1) opredeljuje govorne amplitude za N_{sl} vzorcev - dolg zvočni okvir, ki se konča pri vzorcu m. Naj bo tudi $w[n; m]$ okno, ki se bo uporabljalo za

rezanje izgovorjene "besede" v okvirje, ki običajno trajajo 15-20 ms. Najpogosteje uporabljeno okno je Hammingovo okno, vendar je Hannovo okno tudi pogosto uporabljeno [1]. Če $s[n]$ pomeni amplitudo govora pri vzorcu n, potem $x[n;m] = s[n]w[n;m]$. Različne značilke prispevajo različno stopnjo uspešnosti. MFCC-ji včasih s svojo prvo (delta) in/ali drugo (delta-delta) razliko so najpogostejše funkcije. Linijski spektralni pari (ang. *Line Spectral Pairs*) se prav tako zelo pogosto uporabljajo. Druge pogosto uporabljene funkcije so: kratkoročna energija (ang. *Short-Time Energy*), stopnja ničelnega prehoda (ang. *Zero-Crossing Rate*) in višina zvoka (ang. *pitch*).

2.1.1 Linearno napovedno kodiranje (LPC)

Linearni napovedni koeficienti (ang. *Linear Prediction Coefficients*) so koeficienti $a_k, k = 1..., P$ racionalnega spektra $\Theta(z)$:

$$\Theta(z) = \frac{G}{1 - \sum_{k=1}^{P} a_k z^{-k}}, \tag{2.2}$$

kjer je G vzorčna pridobitev. G lahko aproksimiramo z ocenjevalci, ki uporabljajo koeficiente a_k in zaporedje avtokorelacijskih koeficientov govornega signala $r_x(k)$ pri zamikov $k = 0, 1, ..., P$, za

$$G \simeq \sqrt{r_x[0] - \sum_{k=1}^{P} r_x[k]a_k} \tag{2.3}$$

za negovorne okvirje in

$$G \simeq \sqrt{\frac{1}{F_0}\left(r_x[0] - \sum_{k=1}^{P} r_x[k]a_k\right)} \tag{2.4}$$

za govorne okvirje, kjer F_0 označuje frekvenco višine tona.

2.1.2 LPC izpeljani kepstralni koeficienti

LPC izpeljani kepstralni koeficienti so definirani kot:

$$b_k = \begin{cases} \ln G & k = 0 \\ a_k + \sum_{i=1}^{k-1} \frac{i}{k} \theta[i]\, a_{k-i} & 0 < k \leq P \\ \sum_{i=k-P}^{k-1} \frac{i}{k} \theta[i]\, a_{k-i} & k > P \end{cases}, \tag{2.5}$$

kjer je $\theta[i]$ inverzna Z-transformacija od $\theta(z)$.

2.1.3 Linearni spektralni pari (LSP)

Naj bo $A(z)$ polinom v imenovalcu od $\theta(z)$. V (2.2), $A(z)$ se lahko razgradi v dveh polinomih reda $(P+1)$:

$$A_1(z) = A(z) + z^{-(P+1)} A(z^{-1}) \tag{2.6}$$

$$A_2(z) = A(z) - z^{-(P+1)} A(z^{-1}) \tag{2.7}$$

2.1.4 Mel-frekvenčni kepstralni koeficienti (MFCC)

Za izračun mel-frekvenčnih kepstralnih koeficientov se najprej določa filter banke s trikotnimi filtri:

$$H_p[k] = \begin{cases} 0 & k < f[p-1] \\ \frac{2(k - f[p-1])}{(f[p+1] - f[p-1])(f[p] - f[p-1])} & f[p-1] \leq k \leq f[p] \\ \frac{2(f[p+1] - k)}{(f[p+1] - f[p-1])(f[p+1] - f[p])} & f[p] \leq k \leq f[p+1] \\ 0 & k > f[p+1] \end{cases}, \tag{2.8}$$

katerih središčne frekvence $[p]$ s $p = 1, ..., P$ so enakomerno razmaknjene v mel-lestvici:

$$f\left[p\right] = \left(\frac{N_{sl}}{F_s}\right) B_{mel}^{-1}\left(B_{mel}(F_l) + p\frac{B_{mel}(F_h) - B_{mel}(F_l)}{P+1}\right) \tag{2.9}$$

F_s je frekvenca vzorčenja v Hz, medtem ko sta F_l in F_h najnižja ter najvišja frekvenca banke v Hz. Tipični vrednosti za F_l in F_h sta 0 Hz ter $F_s/2$ Hz. B_{mel} pa je določen kot

$$B_{mel}(F) = 1125\ln(1 + \frac{F}{700}) \tag{2.10}$$

Naslednji korak je izračun logaritemske energije v rezultatih vsakega filtra:

$$S\left[p\right] = \ln\left[\sum_{k=0}^{N_{sl}-1} \left|X\left[k;m\right]\right|^2 H_p\left[k\right]\right], 0 < p \leq P, \tag{2.11}$$

kjer je $X\left[k;m\right]$ kratkoročna diskretna Fourierjeva transformacija (DFT) od $x\left[n;m\right]$. Na koncu izračunamo diskretno kosinusno transformacijo od P log energije $S\left[p\right]$:

$$c\left[n\right] = \sum_{p=0}^{P-1} S\left[p\right]\cos\left(\pi n\frac{2p-1}{2P}\right), 0 \leq n < P \tag{2.12}$$

2.1.5 Kratkoročna energija (STE)

Kratkoročna energija se izračuna po formuli:

$$STE\left[m\right] = \frac{1}{N_{sl}} \sum_{n=m-N_{sl}+1}^{m} x^2\left[n;m\right] \tag{2.13}$$

2.1.6 Višina tona

Za izračun višine tona (ang. *pitch*) je signal najprej filtriran pri 900 Hz, potem se za vsak okvir uporablja rezanje po naslednji formuli:

$$\hat{x}\left[n;m\right] = \begin{cases} x\left[n;m\right] - C_{thr} & \left|x\left[n;m\right]\right| > C_{thr} \\ 0 & \left|x\left[n;m\right]\right| < C_{thr} \end{cases}, \tag{2.14}$$

kjer C_{thr} predstavlja 30% maksimalne vrednosti $|x[n;m]|$. Izračunamo kratkoročno korelacijo $r[\eta;m]$, kjer je zamik $\eta = 0, 1, ..., N_{sl} - 1$. Za negativne zamike se uporablja soda simetrija avtokorelacije. Višina tona je podana z naslednjo enačbo:

$$\hat{F}_o(m) = \frac{F_s}{2N_{sl}} \arg\max_{\eta} r[\eta;m]_{\eta=N_{sl}(\frac{2F_l}{F_s})}^{\eta=N_{sl}(\frac{2F_h}{F_s})}, \tag{2.15}$$

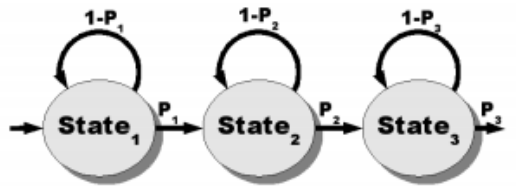
kjer je F_l najnižja frekvenca tona, ki jo človek lahko zazna (običajno 50 Hz), F_h pa je najvišja (običajno 500 Hz). Navadna višina tona je med 65 in 500 Hz ter je zelo odvisna od tega, ali je govorec moški ali ženska.

2.2 Segmentacija

Algoritmi za segmentacijo govorcev so razdeljeni v tri kategorije: algoritmi na osnovi modela, algoritmi na osnovi metrike in hibridni algoritmi, ki kombinirajo oba pristopa [1].

2.2.1 Segmentacija na osnovi modela

V modelski segmentaciji so izpeljani in usposobljeni nabor modelov za različne govorne razrede od korpusa in potem vhodni govor je razvrščen z uporabo teh modelov. To pomeni, da je potrebno predhodno znanje za inicializacijo teh modelov govorcev. En tak osnovni algoritem je univerzalni model ozadja (ang. *Universal Background Model*), ki je vnaprej usposobljen, da ustvari generični govorniški model [1]. V času segmentacije ta model razlikuje med govornimi in negovornimi segmenti. Ker so modeli vnaprej izračunani, se lahko algoritem uporablja v realnem času. Drugi generičen model je model vzorčnih govorcev (ang. *Sample Speaker Model*), ki je vnaprej določen generični model, je neodvisen od govorcev in je zgrajen z vzorčenjem vhodnega zvočnega toka. Modeli so lahko ustvarjeni tudi s pomočjo prikritih Markovskih modelov (ang. *Hidden Markov Models*) ali metode podpornih vektorjev (ang. *Support Vector Machines*). Algoritmi, ki so na osnovi modela, dosežejo



Vir: *Marijn Huijbregts [2]*

Slika 2.1: Tipična HMM-topologija

zmerno stopnjo priklica in visoko stopnjo preciznosti.

2.2.1.1 Prikriti Markovski modeli (HMM)

Statistični model, ki se najbolj pogosto uporablja za izračun verjetnosti $P(O|W)$, je prikriti Markovski model [2]. HMM je sestavljen iz končnega števila stanj, ki so povezani v fiksno topologijo (poglej sliko 2.1). Vhod HMM-ja so značilnostni vektorji, ki se imenujejo opažanja. Vsako HMM-stanje lahko "oddaja"opažanje o_i iz zaporedja opažanj $O = (o_1, o_2, ..., o_T)$ z določeno verjetnostjo, opredeljeno z njeno funkcijo verjetnostne porazdelitve (ang. *Probability Distribution Function*). Prvo opažanje mora biti oddano iz stanja, ki je definirano kot eno izmed začetnih stanj. Ko se to opažanje predela, je eno izmed stanj, ki je povezano z začetnim stanjem, izbrano, da odda naslednje opažanje. Verjetnost, da je izbran en prehod iz enega stanja v drugega, je modelirana z verjetnostjo prehoda. Vsota vseh odhodnih tranzicijskih verjetnosti vsakemu stanju bi morala biti ena, da je skupni verjetnostni prehod tudi ena. Sčasoma so vsa opažanja oddana od stanja, ki je povezano s stanjem, ki je oddalo prejšnje opažanje, in na koncu bi eno izmed zadnjih stanj oddalo opažanje o_T. S premikanjem skozi več različnih poti skozi modele se lahko ustvarijo identična zaporedja opažanj. Dejanska pot, ki je potrebna, da se ustvari določeno zaporedje, ni znana opazovalcu in zato je ta vrsta Markovega modela imenovana prikriti Markov model [2].

2.2.1.2 Mešanice Gaussovih porazdelitev (GMM)

Funkcije verjetnostne porazdelitve za HMM so pogosto mešanice Gaussovih porazdelitev (ang. *Gaussian Mixture Models*) [2]. GMM je zvezna funkcija, oblikovana iz mešanice Gaussovih funkcij, kjer je izhod vsakemu Gaussu pomnožen z določeno utežjo w. Gaussove uteži se seštevajo v 1 in same Gaussove funkcije so opredeljene z njihovim srednjim vektorjem μ ter s kovariančno matriko Σ. Kovariančna matrika Σ je večinoma omejena v diagonalni obliki, ker kot takšna bistveno poenostavlja proces dekodiranja in učenja. Naslednja formula definira GMM z i vhodnih Gaussovih vektorjev n dimenzij, kjer je $(o - \mu_i)^T$ transponent $(o - \mu_i)$:

$$f(o) = \sum_i w_i \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} e^{-\frac{1}{2}(o-\mu_i)\sum_i^{-1}(o-\mu_i)^T} \tag{2.16}$$

Povprečni vektor, kovariančna matrika in utež v vsakem Gaussu v GMM morajo biti nastavljeni tako, da je $f(o)$ največja pri razredu opažanja, ki predstavlja GMM. Ta optimizacija se izvede sočasno z optimizacijo verjetnosti prehoda HMM-ja z uporabo maksimizaciji pričakovanj (ang. *Expectation Maximization*).

2.2.1.3 Viterbijev algoritem

Viterbijev algoritem se uporablja za reševanje problema dekodiranja HMM-ja pri iskanju najbolj verjetne poti, če je dano zaporedje opažanj [2]. Viterbijev algoritem tudi zagotavlja točno verjetnost, če je dana najbolj verjetna pot.

Viterbi z uporabo matrike

Optimalna pot skozi HMM in njena ustrezna ocena sta lahko izračunani z matriko z eno vrstico za vsako stanje s v HMM in en stolpec za vsak časovni interval t. Vsak element v matriki bo imel dve informaciji: največja verjetnost, da je v stanju s v času t in stanju v času $t - 1$, iz katerega je bil sprejet prehod za pridobitev te največje verjetnosti. Matrika je izpolnjena

na naslednji način: najprej so vsi elementi v stolpcu t_0, ki ustrezajo enemu od začetnih stanj, nastavljeni na 1, preostali pa na 0. Nato se vsaka celica v naslednjem stolpcu zapolni z novim rezultatom v:

$$v(s_x, t_i) = \arg\max_S v(S, t_{i-1}) P_S(o_t) P_t(S, s_x), \tag{2.17}$$

kjer je P_S PDF iz stanja S in $P_t(S, s_x)$ prehodna verjetnost iz stanja S do stanja s_x. Število stanj, ki je povzročilo največje število točk, je tudi shranjeno. Zadnji stolpec, ki predstavlja končni časovni okvir T, vsebuje vse verjetnosti po oddaji vseh opažanj. Največja vrednost vseh celic, ki ustreza eni izmed končnih stanj, je verjetnost najverjetnejše poti. Sama pot je pridobljena z vračanjem nazaj čez vsa stanja do prvega stolpca.

Viterbi s podajanjem žetona

Priročen način izračuna rezultata Viterbi je s podajanjem žetona. V tem algoritmu je vsako stanje v HMM zmožno hraniti žeton, ki vsebuje optimalno verjetnost, da so v tem določenem stanju v določenem času t_i. V t_0 le začetna stanja bodo imeli žetona. Na vsakem časovnem okvirju, teh žetonov z začetno vrednostjo 1 se bodo prenesli na vseh povezanih stanj. Pred tem se vrednost žetona posodablja in stanje, iz katerega je prišel, se označi na žetonu. Če ima stanje več kot en odhodni prehod, bo žeton razdeljen in se prenese v vsa povezana stanja. Nova vrednost žetona v v stanju s_y, ki prihaja od stanja s_x, bo:

$$v(s_y) = v(s_x) P_{sx}(o_t) P_t(s_x, s_y), \tag{2.18}$$

kjer je P_{sx} PDF stanja s_x in $P_t(s_x, s_y)$ prehodna verjetnost iz stanja s_x v s_y. Možno je, da dva žetona prispeta na istem stanju hkrati. V tem primeru se obdrži žeton z najvišjo vrednostjo in se drugi zavrže. V času T je od vseh žetonov, ki so v enem od končnih stanj, izbran žeton z najvišjo vrednostjo. Vrednost tega žetona je verjetnost, da je najverjetnejša pot skozi HMM. Dejanska pot pa je označena na samem žetonu.

2.2.2 Segmentacija na osnovi metrike

Segmentacija, ki temelji na metriki, ocenjuje podobnost med sosednjimi analitičnimi okni, premaknjeni v zvočnem posnetku z uporabo neko funkcijsko razdaljo. Lokalni maksimumi funkcijske razdalje, ki presegajo določen prag, se obravnavajo kot točke sprememb. Analitična okna se lahko prekrivajo ali ne, odvisno od uporabe. Te metode ne zahtevajo nobenega predznanja o številu govorcev, njihovi identiteti ali značilnosti signalov. Lahko se uporablja veliko različnih metrik razdalj, pogosto uporabljena metrika pa je Kullback-Leiblerova divergenca ali Gaussova divergenca, občasno pa se uporablja tudi izguba entropije [1]. Alternativno se lahko uporablja generalizirano razmerje verjetnosti (ang. *Generalized Likelihood Ratio*). Najbolj priljubljen kriterij je Bayesov informacijski kriterij (ang. *Bayes Information Criterion*) [1] [2] [4]. Algoritmi, ki delajo segmentacijo na osnovi metrike, prinesejo visoko stopnjo priklica in zmerno stopnjo preciznosti.

2.2.2.1 Bayesov informacijski kriterij (BIC)

BIC je bil dobljen z upragovanjem GLR-ja in je asimptotično optimalen Bayesov kriterij [1] [4], ki se uporablja za odločitev, katere od N_c parametričnih modelov predstavljajo najboljši M podatkovnih vzorcev $x_i \in R^d, i = 1, 2, ..., M$. Vzorci x_i so vektorji dimenzije d in predpostavlja se, da so neodvisni. Za segmentacijo govorcev, le dva različna modela sta uporabljena ($N_c = 2$). Kadar sta dve sosednji analitični okni X in Y okoli časa t_j, je naloga, da določi točko, pri kateri pride do spremembe govorcev v času t_j. Naj bo $Z = X \cup Y$. Problem je formuliran kot statistični test med dvema hipotezama. Pod H_0 ni sprememb govorcev v času t_j. Podatkovni vzorci v Z so oblikovani z večvariantno Gaussovo PDF, katere parametri so povprečni vektor in kovariančna matrika. Naj θ_Z označuje omenjene parametre. θ_Z se lahko oceni z največjo verjetnostjo (ang. *Maximum Likelihood*) ali z uporabo robustnih ocenjevalcev, kot je M-ocenjevalec. Logaritemska verjetnost L_0 se izračuna kot [1]:

$$L_0 = \sum_{i=1}^{N_X} \log p(x_i|\theta_Z) + \sum_{i=1}^{N_Y} \log p(y_i|\theta_Z), \tag{2.19}$$

kjer so N_X in N_Y števila podatkovnih vzorcev v analiznih oknih X in Y. Pod H_1 nastopi točka spremembe govorcev v času t_j. Analizni okni X in Y sta modelirani od različnih večvariantnih Gaussovih gostot, katerih parametri so označeni kot θX in θY. Potem se logaritemska verjetnost L_1 izračuna kot:

$$L_1 = \sum_{i=1}^{N_X} \log p(x_i|\theta_X) + \sum_{i=1}^{N_Y} \log p(y_i|\theta_Y) \tag{2.20}$$

Različnost med dvema sosednjima analiznima oknoma X in Y z BIC je ocenjena kot:

$$\delta = L_1 - L_0 - \frac{\lambda}{2}\left(d + \frac{d(d+1)}{2}\right)\log N_Z, \tag{2.21}$$

kjer je $N_Z = N_X + N_Y$ število vzorcev v oknu Z. λ je podatkovno odvisen kazenski faktor (idealno 1). Če $\delta > 0$ se poiščeta lokalni maksimum δ in čas t_j, se obravnava kot sprememba govorcev. Če $\delta < 0$ ne obstaja sprememba govorcev v času t_j.

2.2.3 Hibridna segmentacija

Hibridni algoritmi združujejo tehnike na osnovi metrike in na osnovi modela. Ponavadi se najprej uporablja segmentacija na osnovi metrike, da se presegmentira vhodni zvočni signal. Dobljeni segmenti se uporabljajo za ustvarjanje modelov govorcev in potem resegmentacija na osnovi modela rafinira prejšnjo segmentacijo [1].

2.2.4 Vrednotenje uspešnosti segmentacije

Dva para enačb se pogosto uporabljata za oceno uspešnosti segmentacije. Po eni strani se lahko uporablja stopnja napačnih pozitivnih primerov (ang.

False Alarm Rate) in stopnja neuspešnega odkrivanja (ang. *Miss Detection Rate*) [1], ki sta definirani kot:

$$FAR = \frac{FA}{GT + FA}, \tag{2.22}$$

$$MDR = \frac{MD}{GT}, \tag{2.23}$$

kjer FA označuje število lažnih alarmov, MD število neuspešnih odkrivanj in GT predstavlja dejansko število sprememb govorcev. Lažni alarm se pojavi takrat, ko je zaznana točka spremembe, čeprav ne obstaja. Neuspešno odkrivanje pride, ko obstoječa točka spremembe ni zaznana.

Po drugi strani se lahko uporabljajo natančnost (ang. *precision*), priklic (ang. *recall*) in ocena F_1:

$$PRC = \frac{CFC}{DET} = \frac{CFC}{CFC + FA}, \tag{2.24}$$

$$RCL = \frac{CFC}{GT} = \frac{CFC}{CFC + MD}, \tag{2.25}$$

$$F_1 = 2\frac{PRC \cdot RCL}{PRC + RCL}, \tag{2.26}$$

kjer CFC označuje število pravilno ugotovljenih sprememb in $DET = CFC+FA$ je število ugotovljenih sprememb. F_1 ima vrednost med 0 in 1, čim večja je njena vrednost, bolj uspešen je algoritem.

Med pari veljajo naslednje enačbe [2]:

$$MDR = 1 - RCL, \tag{2.27}$$

$$FAR = \frac{RCLFA}{DET \cdot PRC + RCL \cdot FA}. \tag{2.28}$$

2.3 Gručenje

Pristopi za gručenje govorcev so razdeljeni v dve glavni kategoriji: deterministični in verjetnostni [1]. Deterministični pristopi gručijo skupaj podobne zvočne segmente glede meritve, medtem ko verjetnostni pristopi uporabljajo GMM ali HMM, da modelirajo gruče.

2.3.1 Deterministične metode

Deterministične metode gručijo skupaj podobne zvočne segmente. Pogledali bomo SOM-metode in hierarhične metode.

2.3.1.1 Metode, ki temeljijo na SOM

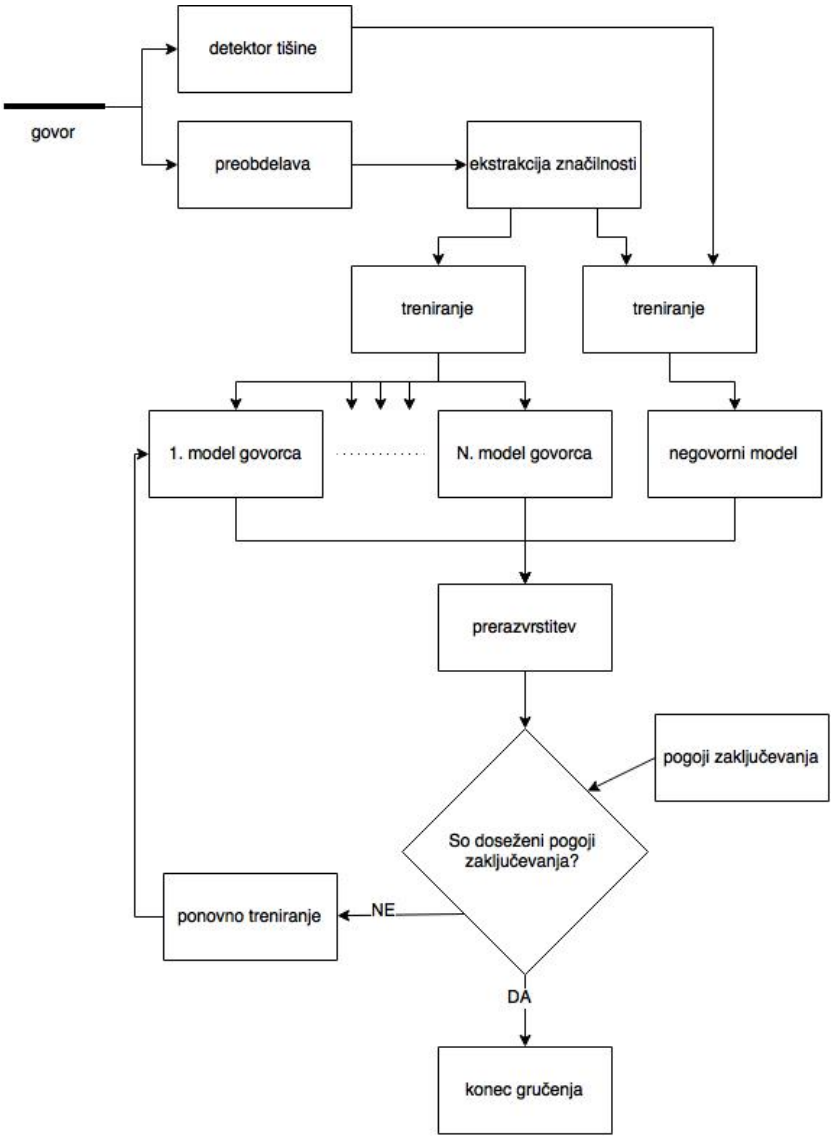
Samoorganizirajoče mreže[1] (ang. *Self-organising maps*) so močno orodje za gručenje govorcev, vendar se predpostavlja, da je število govorcev vnaprej znano. Podatki so razdeljeni v kratkih segmentih in predpostavlja se, da ima vsak segment samo en govorec in da je dovolj dolg (običajno 0.5 s). Uporablja se več SOM-ov in po segmentaciji na govorne ter negovorne segmente s pomočjo upragovanja se uporabljajo za učenje SOM-a. Najbolj pogosto je vsak govorec modeliran s Kohonenova SOM od 6 X 10 nevronov [1]. Na sliki 2.2 je splošen opis enega takega sistema.

2.3.1.2 Hierarhične metode

Predpostavlja se, da ima vsak segment samo en govorec. Najbližje pare zvočnih segmentov najdemo s primerjavo razdalj med vsemi segmenti, ki jih imamo na voljo. Za izračun razdalje med segmentoma s_i in s_j se uporablja GLR [1]:

$$GLR(s_i, s_j) = \frac{L(s_c; \mu_c, \Sigma_c)}{L(s_i; \mu_i, \Sigma_i)L(s_j; \mu_j, \Sigma_j)}, \tag{2.29}$$

[1]Samoorganizirajoča mreža je ena izmed najbolj priljubljenih modelov nevronskih mrež. Spada v kategorijo konkurenčnih učnih mrež in temelji na nenadzorovanem učenju.



Vir: *Margarita Kotti [1]*

Slika 2.2: Nenadzorovan sistem za gručenje govorcev

kjer so $s_c = s_i \cup s_j$ in μ_c, μ_i in μ_j povprečne vektorske lastnosti v zvočnih segmentih. Σ_c, Σ_i in Σ_j so kovariančne matrike in L je verjetnost podatkov. Poleg tega se uporablja notranjo-gručna disperzija:

$$G(c) = |\sum_{i=1}^{N_c} n_i \Sigma_i| \sqrt{N_c}. \tag{2.30}$$

2.3.2 Verjetnostne metode

Verjetnostne metode uporabljajo GMM ali HMM, da zgradijo modele, ki opisujejo gruče [1].

2.3.2.1 Metode, ki temeljijo na GMM

Naj bodo $\{s_1, s_2, ..., s_N\}$ N neopredeljeni zvočni segmenti. Vsak segment je izgovoril eden od N_s govorcev. Referenčni prostor se lahko ustvari bodisi z uporabo N segmentov, ki se gručijo, ali pa od drugega arbitrarnega govora nabor podatkov, sestavljen iz K baz. Vsaka baza se nanaša na reprezentativne govorne značilke. Potem ko je izdelan referenčni prostor, je vsak segment preslikan v K-dimenzionalni projekcijski vektor $V_i = v(s_i, \phi_1), v(s_i, \phi_2), ..., v(s_i, \phi_K)]^T$, kjer $v(s_i, \phi_k)$ označuje, koliko segmenta s_i lahko označimo z vektorjem ϕ_k. Če dva segmenta s_i in s_j izvirata iz istega govorca, bo večina projekcijskih vrednosti V_i in v_j podobna. Podobnost je izračunana s kosinusno podobnostjo (ang. *Cosine Similarity Measure*):

$$CSM(s_i, s_j) = \frac{V_i^T V_j}{||V_i|| ||V_j||}. \tag{2.31}$$

Segmenti, ki so dovolj podobni, so združeni v gručo.

Gaussovo modeliranje, ki je za posamezno izrekanje, uporablja en GMM za vsak N segment, ki se gruči [1]. Te N GMM-e, $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_N$ tvorijo referenčne baze $\phi_k = \theta_k, k = 1, 2, ..., N$. Za vsak segment s_i je njegova projekcija na bazi ϕ_k izračunana z:

$$v(s_i, \phi_k) = \log p(s_i|\theta_k). \tag{2.32}$$

Gaussovo modeliranje, ki je za univerzalno izrekanje, uporablja en šifrant za neodvisno izrekanje, ki ima K kodnih besed. Šifrant je univerzalni model, ki je usposobljen za kritje govorčeve neodvisne distribucije značilnostnih vektorjev. Vsaka kodna beseda $kb, k = 1, 2, ..., K$ je sestavljena iz poprečnega vektorja μ_k in diagonalne kovariančne matrike Σ_k. Učenje šifranta se izvaja z uporabo k-means[2] in Mahalanobove razdalje [1]. Potem ko se ustvarijo K kodne besede, se na vsakem značilnostnem vektorju dodeli indeks kodne besede. Projekcijska vrednost $v(s_i, \phi_k)$ se izračuna kot:

$$v(s_i, \phi_k) = \frac{\#\text{ značilnostnih vektorjev v } s_i, \text{ dodeljeni na } kb_k}{\#\text{ značilnostnih vektorjev v } s_i}, \tag{2.33}$$

kjer # označuje kardinalnost niza.

Gaussovo modeliranje, ki je za univerzalno izrekanje, in po tem še prilagoditev modela skupaj ustvarja univerzalen GMM z uporabo vseh segmentov, ki se gručijo.

2.3.2.2 Metode, ki temeljijo na HMM

HMM se pogosto uporablja za gručenje govorcev [1]. Vsako stanje na HMM predstavlja gručo in PDF vsake gruče je modeliran z GMM. HMM je usposobljen s pomočjo EM-algoritma. Inicializacija PDF-ja se opravi z uporabo k-means algoritma. Tehnika se začne z gručenjem podatkov, več kot je potrebno (večje število gruč) s ciljem, da se zmanjša verjetnost, da so različni govorci gručeni v enem razredu. Nato se segmentacija izvede z uporabo Viterbijevega algoritma v vsakem razredu. Naslednji korak je zmanjšanje število razredov z združitvijo. Skupine so združene z določeno razdaljo, kot je na primer GLR. Nov razred je predstavljen od drugega GMM, ki ima enako število komponent kot vsoto komponent posameznih razredov. Parametri te

[2]Popularen algoritem za gručenje

novonastale gruče so spet usposobljeni z EM-algoritmom z uporabo značilk, ki spadajo v gruče, ki se združujejo. Segmentacija je ponovno ocenjena z novo HMM-topologijo, ki ima en grozd manj, in se izračuna verjetnost podatkov na tej segmentaciji. Verjetnost narašča, če so podatki v tej gruči istega govorca. V nasprotnem primeru se zmanjša, če so od različnih govorcev. Proces združevanja se ustavi, ko se verjetnost ne zmanjšuje več.

Poglavje 3

Orodja za diarizacijo govorcev

3.1 LIUM Speaker diarization

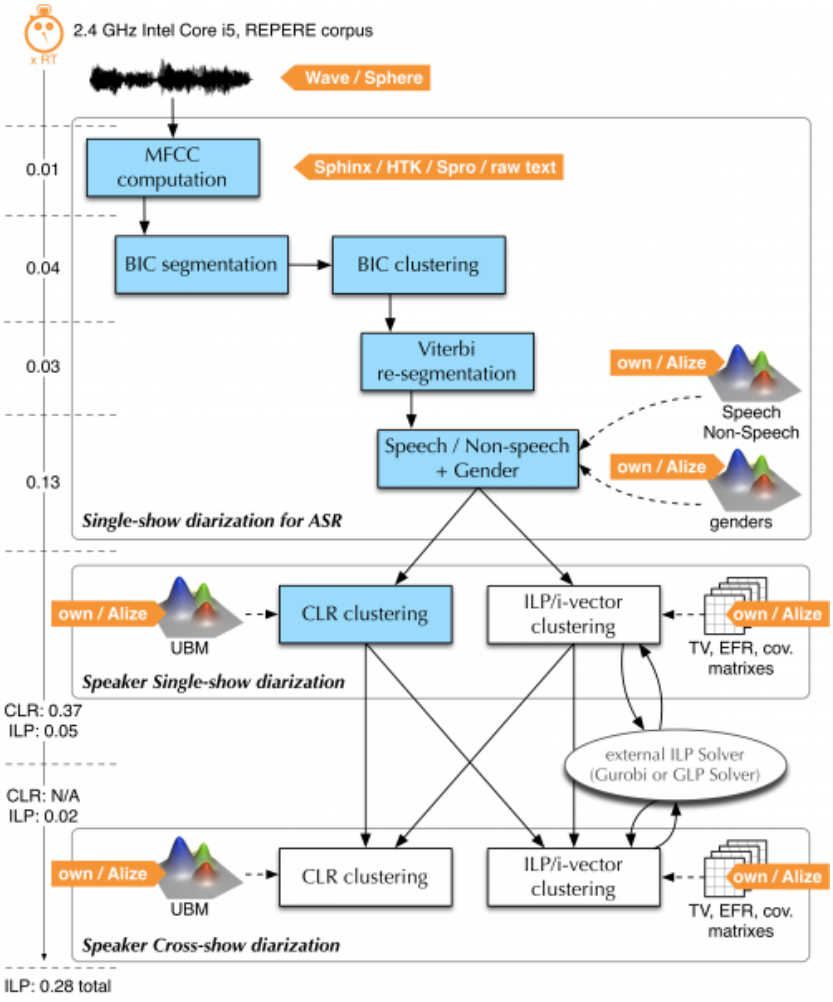
LIUM_SpkDiarization [5] (v nadaljevanju LIUM) je programska oprema, namenjena za diarizacijo (segmentacija in gručenje) govorcev in je napisana v Javi. LIUM obsega celoten nabor orodij za ustvarjanje popolnega sistema za diarizacijo govorcev - od zvočnega signala do gručenje govorcev na podlagi CLR-meritve. LIUM vključuje MFCC-računanje, odkrivanje govora in metod za diarizacijo govorcev. To orodje je optimizirano za radijske ali televizijske oddaje, vendar lahko se prilagodi tudi za druge posnetke. Slika 3.1 prikazuje klasičen postopek za diarizacija govorcev z uporabo LIUM-a. Čas izračuna v vsakem koraku je prikazan na levi strani slike.

LIUM uporablja BIC-algoritem, ki temelji na hierarhičnem aglomerativnem[1] gručenju [5]. Algoritem izvaja dekodiranje Viterbi za ustvarjanje nove segmentacije. Gruča je modelirana s HMM s samo enim stanjem, zastopana z GMM 8 komponent (diagonalna kovarianca). Algoritem ima tudi detekcijo spola, za katero naloga uporablja GMM s 128 diagonalnimi komponentami.

Izvedba datoteke JAR[2] neposredno sklicuje diarizacijo govorcev. Denimo, da moramo narediti diarizacijo "A_01.seg"na zvočno datoteko "A_01.wav",

[1] ang. *agglomerative* proces zbiranja v masi, gruča ponavadi različnih elementov

[2] ang. *Java Archive*, javanska zbirka



Vir: *http://www-lium.univ-lemans.fr/diarization/doku.php/overview [5]*

Slika 3.1: Diarizacija govorcev z LIUM

ukazna vrstica bi bila:

```
/usr/bin/java -Xmx2024m -jar ./LIUM_SpkDiarization.jar \
--fInputMask=./A_01.wav --sOutputMask=./A_01.seg \
--doCEClustering A_01.seg
```

- Možnost –fInputMask=./A_01.wav je ime zvočne datoteke. Lahko je v formatu WAV ali Sphere, tip je avtomatično zaznan glede končnice datoteke.
- Možnost –sOutputMask=./A_01.seg je izhodna datoteka, ki vsebuje segmentacijo.
- Če je možnost –doCEClustering nastavljena, program preračuna CLR gruče na koncu. Stopnja napake je čim bolj minimizirana. Če ta možnost ni nastavljena, se program ustavi takoj po odkritju spola in dobljena segmentacija zadostuje za transkripcijski sistem.
- A_01.seg je ime zvočnega posnetka.

Dodatne možnosti so:

- –trace za prikaz informacij med predelavo;
- –help za prikaz kratkih navodil o uporabi teh orodij;
- –saveAllStep shranjuje vsak korak diarizacije;
- –loadInputSegmentation naloži začetno segmentacijo –sInputMask. Privzeto je začetna segmentacija sestavljena iz enega segmenta, ki sega od začetka do konca zvočnega posnetka;
- –thresholds=1.5:2.5,2.5:3.5,250.0:300,0:3.0 nabira nadzor pragov. Pari od leve proti desni so: nastavitev minimalnega in maksimalnega praga za BIC-segmentacijo drugega obrata, korak je 0,5; minimalni in maksimalni prag za BIC-grozdenje, korak je 0,5; minimalni in maksimalni prag za dekodirajoči kazenski prag Viterbi, korak je 50; minimum in maksimum praga za CLR grozdenja, korak je 0,01.

Format izhodne datoteke je naslednji:
A_01 1 0 300 M S U spk0

A_01: ime datoteke;

1: številka kanala;

0: začetek segmenta;

300: dolžina segmenta;

M: spol (M/F);

S: tip (telefon, studio);

U: tip okolja (glasba, govor ...);

spk0: oznaka govorca.

3.2 PyAudioAnalysis

PyAudioAnalysis je odprta knjižnica Python, ki ponuja ekstrakcijo značilk, klasifikacijo, segmentacijo in vizualizacijo zvočnih posnetkov. Koda knjižnice je organizirana v nekaj datotek Python:

audioAnalysis.py: ta datoteka je vmesnik knjižnice in izvaja ukazne vrstice;

audioFeatureExtraction.py: tukaj se izvaja ekstrakcija značilk;

audioTrainTest.py: ta datoteka izvaja zvočne postopke za razvrščanje;

audioSegmentation.py: ta datoteka izvaja zvočne funkcionalnosti;

audioBasicIO.py: ta datoteka izvaja nekaj osnovnih vhodnih in izhodnih zvočnih funkcionalnosti;

audioVisualization.py: ta datoteka ponuja razne vizualizacije.

Obstajata dve fazi v ekstrakciji zvočnih značilk: kratkoročna in vmesna ekstrakcija. Kratkoročna ekstrakcija se izvaja v funkciji stFeatureExtraction(). Ta razdeli vhodni signal v kratkoročna okna in izračuna značilnosti za vsako okno. Ta postopek nam da zaporedje kratkoročnih vektorjev lastnosti za celotni signal. Vmesna ekstrakcija se izvaja v funkciji mtFeatureExtraction(), ki ekstraktira par statističnih podatkov za vsako kratkoročno značilnostno sekvenco. Število takih kratkoročnih značilk, ki so v tem algoritmu je 34.

Segmentacija je lahko nadzorovana ali nenadzorovana. Za nadzorovano segmentacijo se uporablja nadzorovano znanje za razvrščanje in segmentiranje. To je bodisi z uporabo klasifikatorja ali pa s HMM, da bi dosegli skupno segmentacijo in klasifikacijo. Za nenadzorovano segmentacijo (diarizacija govorcev) nadzorovani model ni na voljo in gručimo ugotovljene segmente. Pri diarizaciji, za ekstrakcija značilk se uporabljajo povprečja in standardni odkloni od MFCC, za gručenje pa metodo k-means. Če število govorcev ni vnaprej znano, je proces gručenja ponovljen za različno število govorcev, da bi našli najbolj optimalno število govorcev. Zadnji korak je glajenje, ki uporablja mediansko filtriranje na gruče in glajenje s pomočjo Viterbijevega algoritma. Funkcija speakerDiarization() se uporablja za ekstrakcijo zaporedja zvočnih segmentov in ustrezne oznake gruče.

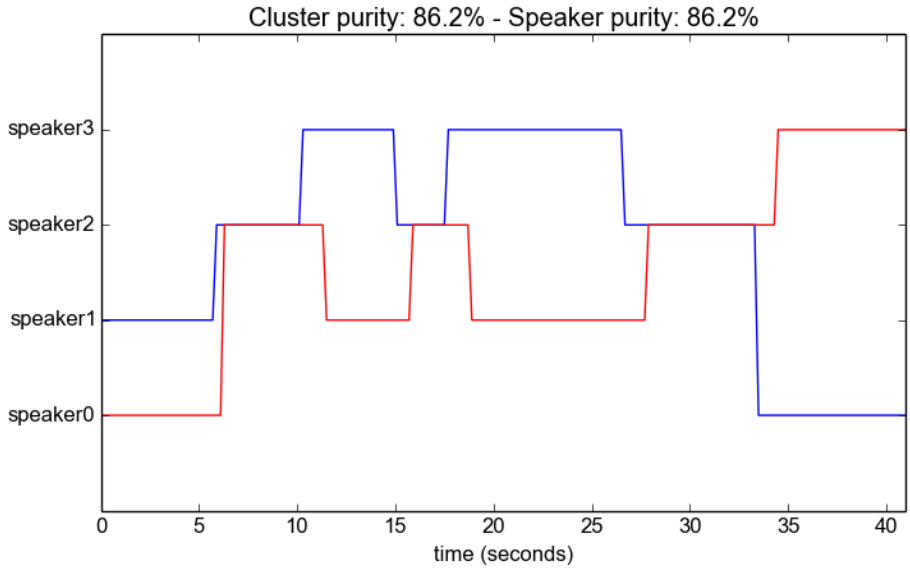
Denimo, da moramo narediti diarizacijo na zvočno datoteko "A_01.wav", ukazna vrstica bi bila:

```
python audioAnalysis.py speakerDiarization -i A_01.wav,
```

če ni znano število govorcev, ali:

```
python audioAnalysis.py speakerDiarization -i A_01.wav \
--num 4,
```

če je število govorcev vnaprej znano. Vizualni primer rezultata je na sliki 3.2, kjer je modra črta rezultat, rdeča pa dejanska resnica (ang. *ground truth*).



Vir: *https://github.com/tyiannak/pyAudioAnalysis/wiki/5.-Segmentation*

Slika 3.2: Vizualni primer rezultata v pyAudioAnalysis

3.3 Diarize-jruby

Ta knjižnica ponuja orodje za segmentacijo govorcev in identifikacijo od drugih zvokov. Ovija binarno JAR-datoteko od LIUM-a in je zgrajena na njegovi osnovi. Dodatne lastnosti so: normalizacija modelov govorcev z uporabo M-Norm, simetrična Kullback-Leibler divergentna aproksimacija in podpora za govorčeve supervektorje, ki se lahko uporabljajo v kombinaciji s to knjižnico Ruby za hitro identifikacijo govorcev.

Primer uporabe:

```
$ jruby -S gem install diarize-jruby
$ jruby -S irb
> require 'diarize'
> audio = Diarize::Audio.new File.read("./A_01.wav")
> audio.analyze!
> speakers = audio.speakers
```

```
> speakers |= other_speakers
> Diarize::Speaker.match(speakers)
```

Format izhodne datoteke je podoben kot pri LIUM-u. Določeni parametri v primeru so skrajšani:

#<Diarize::Segment @speaker_ id=Š0", @duration=2.5, @bandwidth="U", @speaker_ gender="M", @start=0.0, @audio=#<Diarize:: Audio @speakers=..., @segments=[...], @file=#<File: ... /A_ 01.wav>, @path=".../A_ 01.wav", @uri=#<URI::Generic URL: file: ... /A_ 01.wav>, @clusters=...>>

@speaker_ id=Š0": oznaka govorca;

@start=0.0: začetek segmenta;

@duration=2.5: dolžina segmenta;

@speaker_ gender="M": spol (M/F);

@bandwidth="U": tip okolja (glasba, govor ...);

@file=...: zvočna datoteka.

3.4 VoiceID

VoiceID je sistem prepoznavanja in identifikacije, napisan je v Pythonu in temelji na LIUM. VoiceID lahko obdela video- ali zvočne datoteke, da ugotovi, v kateri rezini časa se dogaja diarizacija, in potem analizira te segmente ter opredeljuje, kdo govori. Da naredi to, uporablja podatkovno zbirko o govornih modelih. Možno je uporabljati skripto v "trening način"ali pa se lahko zgradijo ročni modeli govorcev in se dajo v bazo podatkov z uporabo skripte, da bi se ustvarile GMM-datoteke.

Primer uporabe:

Če želimo analizirati video-/zvočne datoteke v interaktivnem načinu:

```
vid -i A\_ 01.wav -u
```

Če želimo analizirati video-/zvočne datoteke v serijskem načinu:

```
vid -i A\_ 01.wav
```

Za izgradnjo glasovnega modela iz dane WAV-datoteke:

```
vid -s SPEAKER_ID -g A\_ 01.wav
```

Če želimo izpisati izhod v JSON-datoteko:

```
vid -i A\_ 01.wav -f json -u
```

Opcija za hrupno zvočno datoteko:

```
vid -i A\_ 01.wav -f json -n -u
```

Izhod je lahko v obliki json[3], srt[4] ali xmp[5].

VoiceID omogoča tudi grafični vmesnik z voiceidplayer (slika 3.3):

```
voiceidplayer video_file database_path
```

Format izhodne datoteke v obliki srt je naslednji:

1
00:00:00,000 –>00:00:05,500
S0

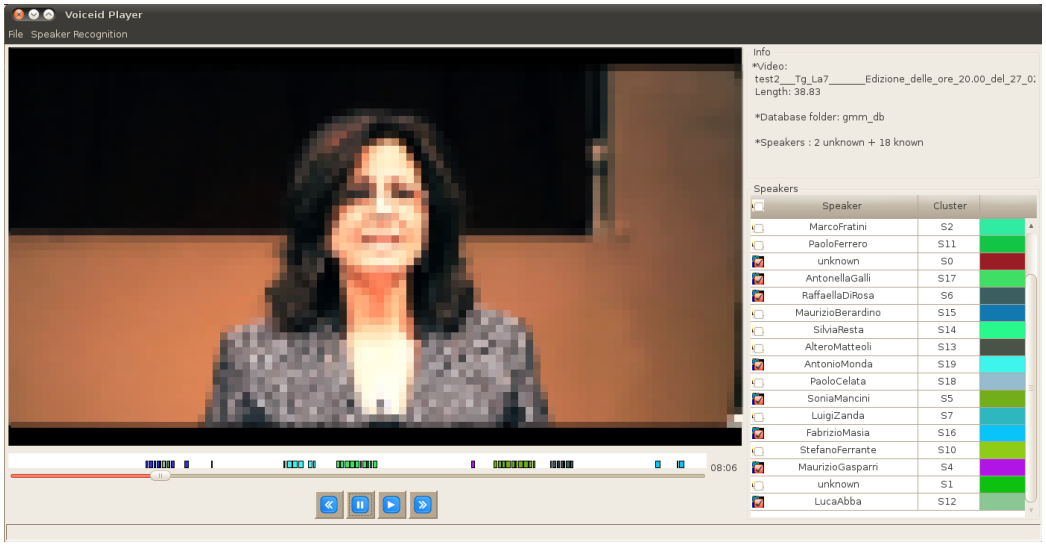
1: zaporedna številka podnapisa;

00:00:00,000: začetek segmenta;

00:00:05,500: konec segmenta;

S0: oznaka govorca.

[3]JavaScript Object Notation - JavaScript notacija za objekte
[4]SubRip datoteka za podnapise
[5]Extensible Metadata Platform



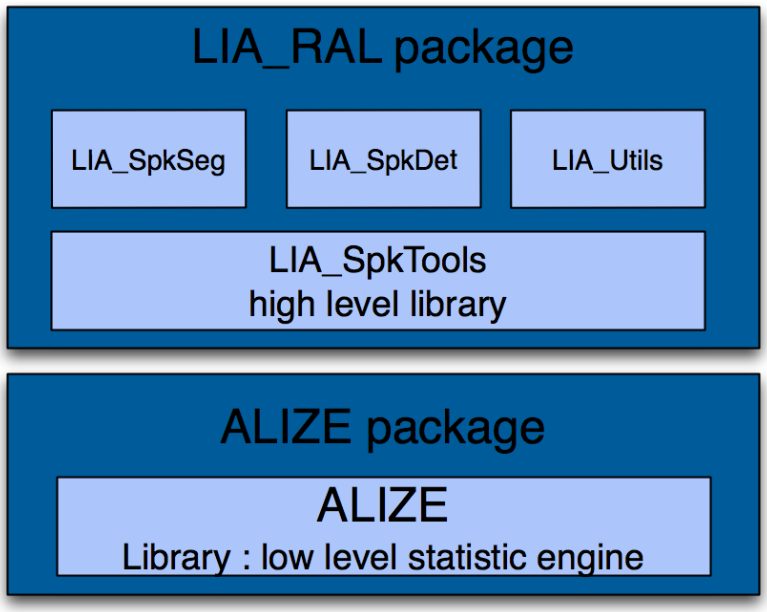
Vir: *https://code.google.com/p/voiceid/*

Slika 3.3: VoiceID z voiceidplayer

3.5 ALIZE

ALIZE je odprtokodna platforma za biometrično avtentikacijo [6] [7]. Cilj ALIZE je omogočiti razvoj biometričnih aplikacij z zagotavljanjem nabora orodij. ALIZE je sestavljen iz več sklopov:

- ALIZE knjižnica, ki je sestavljena iz vseh potrebnih funkcij v zvezi z uporabo GMMje.
- LIA_RAL - orodje, ki je razdeljeno na več delov:
  - LIA_SplTools knjižnica na visoki ravni;
  - LIA_Utils orodja, ki so potrebna za manipulacijo vseh formatov, uporabljenih v ALIZE, GMM- modelov, parametri in podobno;
  - LIA_SpkDet: nabor orodij za vse naloge, ki jih zahteva biometrični sistem za avtentikacijo: modeli, učenje modela, normalizacija parametrov, normalizacija rezultatov in podobno;
  - LIA_SpkSeg, orodje za diarizacijo govorcev.



Vir: *http://mistral.univ-avignon.fr/index_en.html [6] [7]*

Slika 3.4: Struktura ALIZE

Na sliki 3.4 je prikazana struktura ALIZE z vsemi svojimi deli. V tem diplomskem delu bomo pogledali samo te dele, ki so za segmentacijo govorcev (diarizacija). Glavni cilji ALIZE so [6]:

- predlagati orodja, ki bodo naredili hitrejši razvoj nove ideje s state-of-the-art[6] zmogljivosti;
- spodbujati laboratorije za ocenjevanje novih predlogov z uporabo tega orodja;
- pomagati razumevanje algoritmov, kot so EM, Viterbi in podobno;
- preizkusiti nove komercialne aplikacije;
- olajšati izmenjave in prenos znanja med akademskimi laboratoriji ter med laboratoriji in podjetji.

[6]najnovejša faza v razvoju izdelka, ki vsebuje najnovejše ideje in posodobljene lastnosti

ALIZE je razvit v C++. Splošna arhitektura orodja temelji na razdelitvi funkcij med več programskih strežnikov [6] [7]. Glavni strežniki so strežnik značilk, ki upravlja z akustičnih podatkov, mešani strežnik, ki se ukvarja z modeli (skladiščenje, predelava, vezanje sestavnih delov, shranjevanje, branje ...), in statistični strežnik, ki izvaja vse statistične izračune (EM-statistične ocene, verjetnostno računanje, Viterbijevo poravnavo in podobno). Ta arhitektura ponuja več prednosti:

- Vsak strežnik je dostopen, zahvaljujoč zelo kratkim seznamom funkcij na visoki ravni, in te funkcije, ki so na nizki ravni kot upravljanje pomnilnika, so prikrite za uporabnika.
- Vsak strežnik se optimizira in posodablja ločeno.
- Več instanc istega strežnika lahko teče istočasno.
- Uporabnikov kod predstavlja isto strukturo, organizirano med glavnimi strežniki, kar pomaga pri razvoju izvorne kode in razumevanja.
- Arhitektura programske opreme omogoča enostavno distribucijo strežnikov na različnih računalnikih: več primerov istega strežnika lahko teče na različnih računalnikih za povečanje računske moči.

Format izhodne datoteke je naslednji:
1.210000 4.810000 speech0_L0

1.210000: začetek segmenta;

4.810000: konec segmenta;

speech0_L0: oznaka govorca.

3.6 Matlab Audio Analysis Library

Matlab Audio Analysis Library (v nadaljevanju MAAL) je knjižnica Matlab, ki pokriva širok spekter zvočnih analitičnih nalog, kot so splošno zvočno ravnanje (vhod, izhod, predvajanje, snemanje ...), zvočna obdelava, ekstrakcija

značilk, klasifikacija, segmentacija in iskanje informacij v glasbi [8]. Osrednje metode in algoritmi knjižnice so:

- snemanje in predvajanje zvočnega signala;
- zvočno filtriranje;
- zvočna obdelava;
- FFT v kratkem času;
- kratkoročna energija, ničelna stopnja prehoda (ang. *zero crossing rate*), entropija energije;
- vektor Chroma, spektralni lastnosti;
- višina tona, osnovna frekvenca;
- zvočna klasifikacija segmentov, klasifikator k-najbljižji sosed, SVM, odločitvena drevesa;
- zvočna segmentacija;
- HMM, dinamično programiranje;
- povzetek glasbe, glasbena vizualizacija, zmanjšanje dimenzionalnosti.

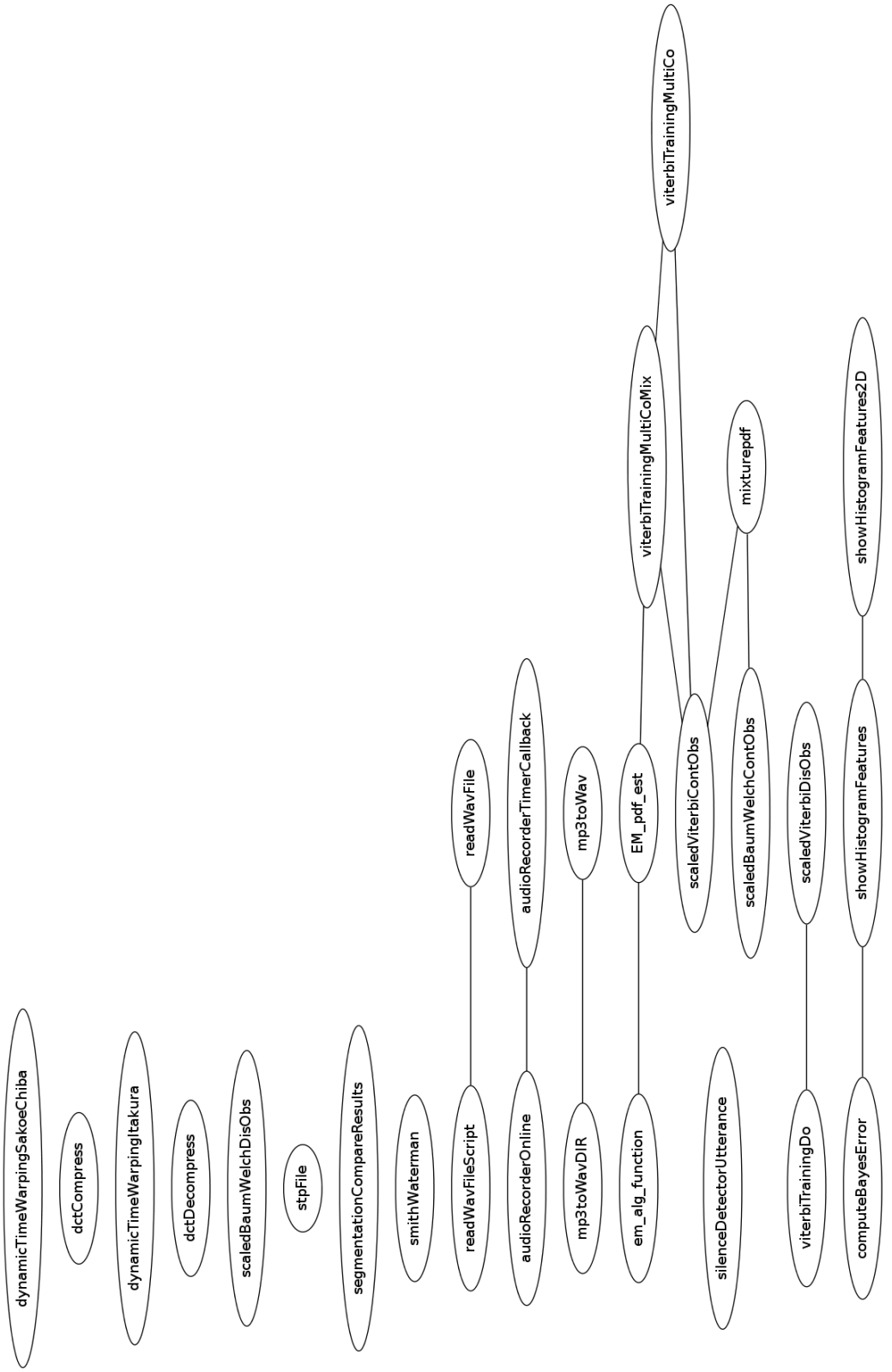
Graf odvisnosti datoteke .m v knjižnici je prikazan na sliki 3.5 in sliki 3.6 [8].

Format izhodne datoteke je:
14 22 1

14: začetek segmenta;

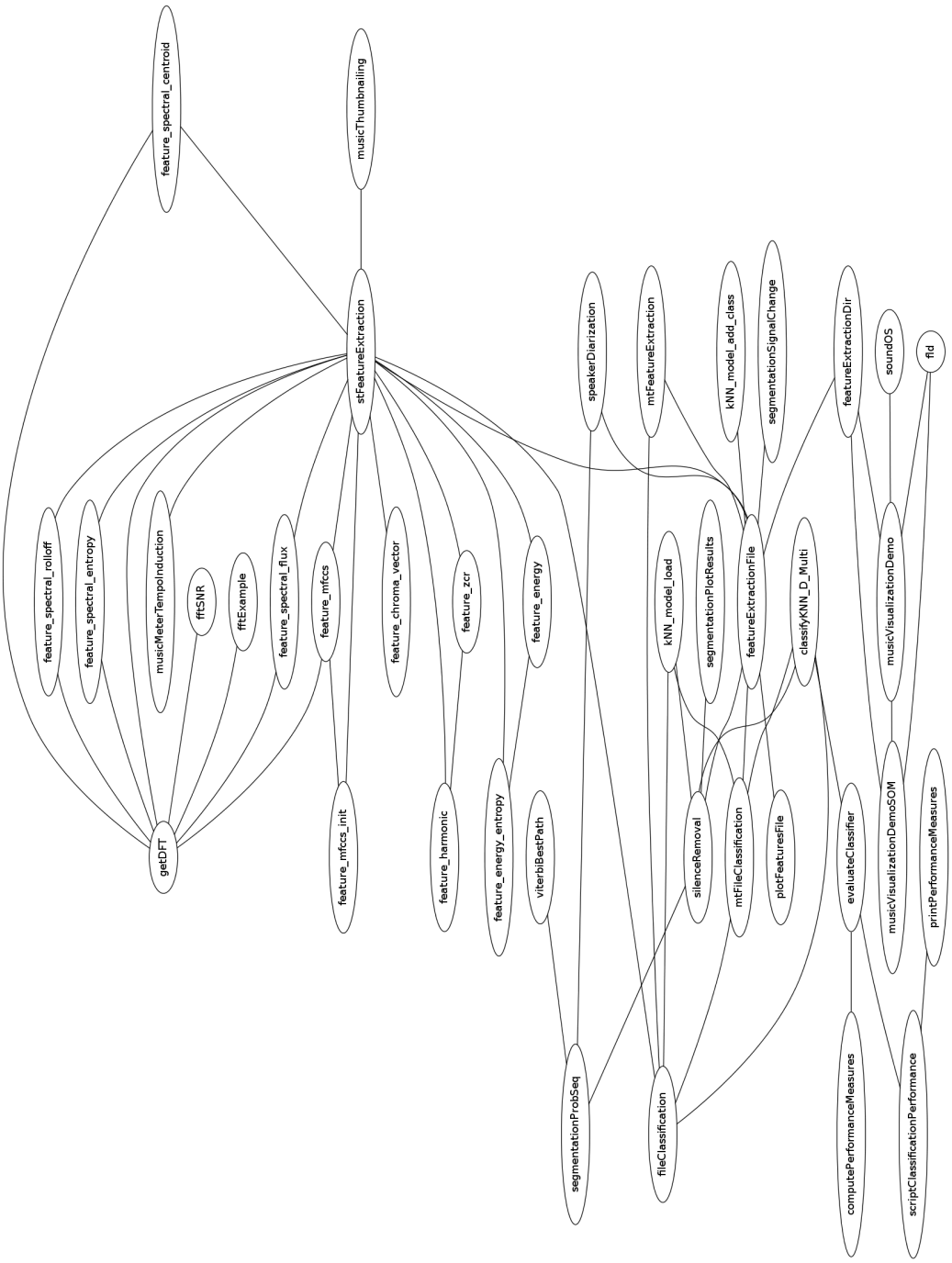
22: konec segmenta;

1: oznaka govorca.



Vir: *[8]*

Slika 3.5: Graf odvisnosti MAAL



Vir: *[8]*

Slika 3.6: Graf odvisnosti MAAL

Poglavje 4

Evalvacija in rezultati

Za evalvacijo knjižnic in algoritmov za diarizacijo govorcev, omenjenih v poglavju 3, je bil, napisan program v programskem jeziku Java. Zvočni posnetki, na katerih so bili ti programi testirani, so bili odrezki nekaj posnetkov iz velike zbirke terenskih posnetkov. Posnetki so iz različnih okolij, imajo različno število govorcev, različno razmerje med govorčevo časovno dolžino govora, drugačno čistost glasov in jasnosti posnetkov ter različno dolžino prekrivanja med govorcem.

4.1 Ročna diarizacija testnih posnetkov

Zaradi velikih imen zvočnih datotek bomo v tem diplomskem delu uporabili naslednja skrajšana imena za te datoteke:

izvirno ime	novo ime
DAT25x, Veliki trn, 2_6_96, Dol_12_s_01_0-5.wav	A_01.wav
DAT193, Pohorje, Kebelj,Brezje priOpl,Bozje,staj,15-19_5_0202_14_s_01_0-5.wav	C_01.wav
OT863_06_s_01_0-5.wav	D_01.wav
OT972_21_s_0-5.wav	E_01.wav
T849-0101_01_0-5.wav	I_01.wav

T877-1&201_13_s_01_0-5.wav	J_01.wav
T1508-1&201_04_s_0-5.wav	K_01.wav
T1510-3&401_10_s_01_0-5.wav	L_01.wav

Ti posnetki so bili okoli 5% celotnega trajanja vseh terenskih posnetkov. Najprej je bila za vsakega od teh zvočnih posnetkov narejena ročna diarizacija: zapisani so bili število govorcev in časovni intervali za vsakega govorca - od kdaj do kdaj govori v posnetku, zaradi človeških omejitev je bilo zaokroženo na najbližjih $25ms$. Ker imajo vsi algoritmi različne izhodne formate rezultatov, je bil uporabljen skupen zapis vseh algoritmov:

0 625 S0

0: začetek segmenta;

625: konec segmenta;

S0: oznaka govorca.

Podatki, kot so spol, okolje posnetka in podobne dodatne funkcionalnosti, ki so jih nekateri algoritmi imeli, se ne upoštevajo. Nekateri algoritmi niso zapolnili celotnega razpona zvočnega posnetka pri njihovem rezultatu, tako da so te luknje izpolnjene z neimenovanim govorcem (šteje se kot napaka, če je na tem intervalu govor). Ti intervali so bili tudi zaokroženi na $25ms$. Število dejanskih govorcev v vsakem posnetku je podano v naslednji tabeli:

ime posnetka	število govorcev
A_01.wav	2
C_01.wav	2
D_01.wav	3
E_01.wav	5
I_01.wav	3

J_01.wav	3
K_01.wav	3
L_01.wav	3

4.2 Vizualizacija ročne diarizacije

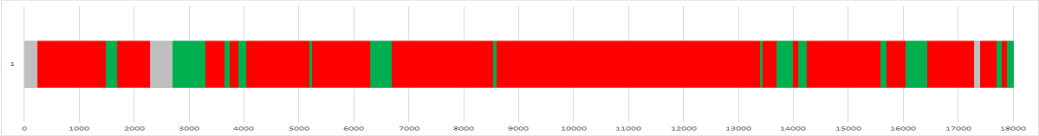
V prejšnjem delu smo pogledali, kako je potekala ročna diarizacija. Na naslednjih slikah je prikazana vizualna segmentacija govorcev. Na osi x je prikazan čas v ms, različne barve pa prikazujejo različne govorce: rdeča - govorec 1, zelena - govorec 2 in tako naprej. S sivo pa je označen del posnetka, v katerem pride do prekrivanja govorcev in ni možno točno določiti 1 govorca za ta časovni interval.

Kratek opis zvočnih datotek:

A_01.wav: Na tem posnetku, ki je prikazan na sliki 4.1, govorita 2 govorca. Je pogovor med moški srednjih let in starejšo gospo. Gospa govori več kot moški in je včasih prekinjena zaradi vprašanja. Obstaja zelo malo prekrivanja med njima. Kakovost posnetka je dobra in vse besede so razumljive.

C_01.wav: Na tem posnetku, ki je prikazan na sliki 4.2, govorita 2 govorca. Je pogovor med žensko in starejšim moškim. Posnetek je dinamičen, pogosto pride do spremembo govorcev. V posnetku je glas moškega dober in razumljiv, glas ženske pa je slabši ter bolj prikrit, vendar razumljiv. Pride do občasnega prekrivanja med njima.

D_01.wav: Na tem posnetku, ki je prikazan na sliki 4.3, govorijo 3 govorci, dve ženski in en moški. Prva gospa govori zelo močno in glasno ter dominira. Glas druge gospe in moškega je manj prevladujoč. Večina časa govori prva gospa, vendar pride do pogostih prekrivanj med njimi.



Slika 4.1: Vizualizacija govorcev zvočnega posnetka A_01.wav

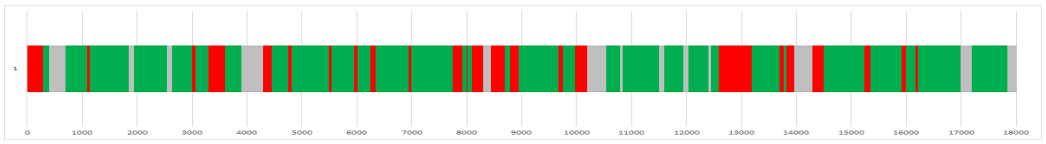
E_01.wav: Na tem posnetku, ki je prikazan na sliki 4.4, govorijo 5 govorci. Starejši moški, starejša ženska in še dve ženski ter moški. Največ časa govori starejša ženska, starejši moški govori le na začetek posnetka, ostale pa občasno vprašajo kakšno vprašanje. Kakovost posnetka je dobra in vse besede so razumljive. Občasno pride do prekrivanja med njimi.

I_01.wav: Na tem posnetku, ki je prikazan na sliki 4.5, govorijo 3 govorci, dva moška in ena ženska. Prvi moški dominira v prvi polovici posnetka, ženska pa v drugi. Obstajajo prekrivanja med njimi. Kakovost posnetka je dobra in vse besede so razumljive.

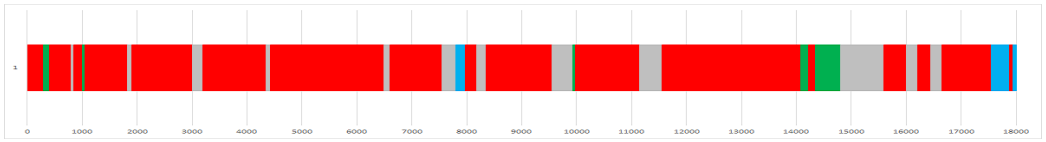
J_01.wav: Na tem posnetku, ki je prikazan na sliki 4.6, govorijo 3 govorci, vendar eden govori večino časa z nekaj prekinitvami ene ženske in enega moškega. Obstaja nekaj prekrivanj med njimi. Kakovost posnetka je dobra in vse besede so razumljive.

K_01.wav: Na tem posnetku, ki je prikazan na sliki 4.7, govorijo 3 govorci, dve ženski in en moški, ki govori v drugi polovici posnetka, v prvi pa je pogovor med tema dvema ženskama. Obstaja veliko prekrivanj med njimi. Kakovost posnetka je dobra, z zelo redkimi motnji iz zunanjega vira.

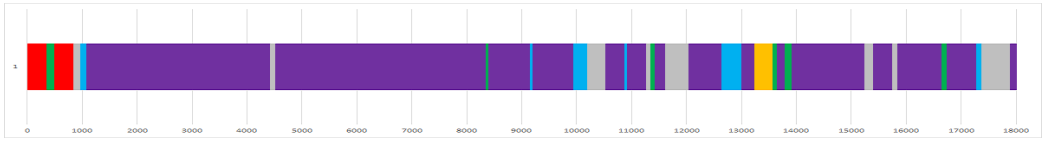
L_01.wav: Na tem posnetku, ki je prikazan na sliki 4.8, govorijo 3 govorci. Govorijo dve ženski in en moški, zelo pogosto se dogaja prikrivanje med njimi, kakovost posnetka pa je dobra.



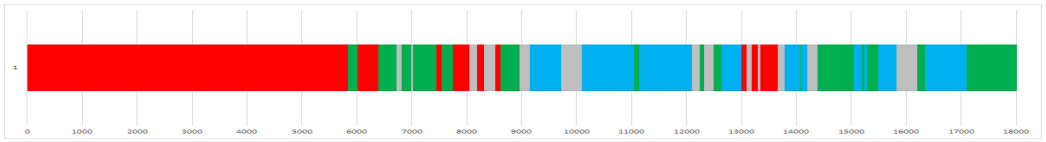
Slika 4.2: Vizualizacija govorcev zvočnega posnetka C_01.wav



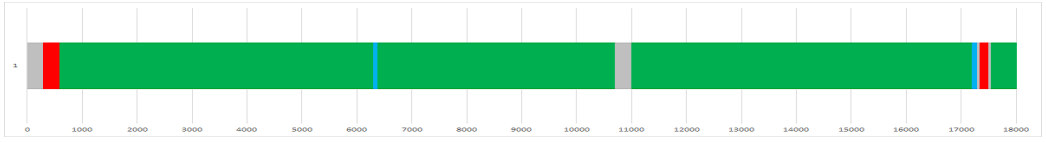
Slika 4.3: Vizualizacija govorcev zvočnega posnetka D_01.wav



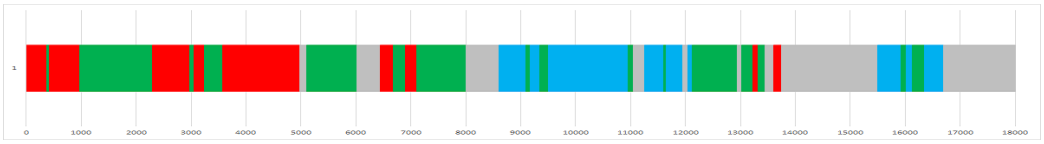
Slika 4.4: Vizualizacija govorcev zvočnega posnetka E_01.wav



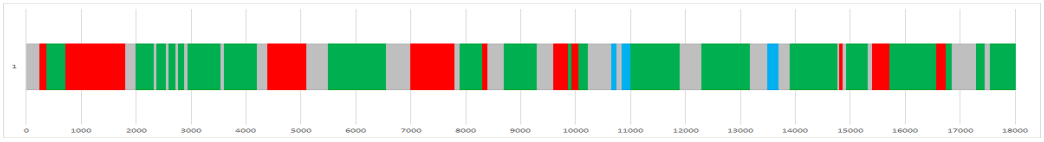
Slika 4.5: Vizualizacija govorcev zvočnega posnetka I_01.wav



Slika 4.6: Vizualizacija govorcev zvočnega posnetka J_01.wav



Slika 4.7: Vizualizacija govorcev zvočnega posnetka K_01.wav



Slika 4.8: Vizualizacija govorcev zvočnega posnetka L_01.wav

4.3 Ocenjevanje algoritmov

Sisteme za diarizacijo govorcev morajo izvesti segmentacija *in* gručenje govorcev. Najbolj pogosto uporabljena metrika, ki se uporablja za ocenjevanje uspešnosti teh algoritmov, je stopnja napak diarizacije (ang. *Diarization Error Rate*) [3] [2]. DER je razmerje nepravilno pripisanega časa govora (napačno zaznan govor, zgrešene detekcije govora in nepravilno gručenje govorcev) s skupnim časom govora. DER se izraža v odstotkih, pri čemer rezultat 0 odstotkov pomeni popolno delovanje, višji odstotki pa kažejo na slabše rezultate. DER je definiran kot [1]:

$$DER = MS + FA + SE. \tag{4.1}$$

MS je zgrešeni govor (ang. *Missed Speech*), ki se pojavi, ko je govorec prisoten, vendar ni zaznan, FA je lažni alarm (ang. False Alarm), ki se pojavi, ko je zaznan, vendar ni prisoten v resnici (Poglej 2.2.4), in SE je napaka govorca (ang. *Speaker Error*), ki se pojavi, ko je zaznan napačen govorec. V tem diplomskem delu bomo uporabljali inverzni DER ($1 - DER$). To povzroča, da je zdaj 100% najboljši rezultat, kar pomeni, da je algoritem zaznal čisto vse spremembe govorcev, točno ugotovil njihovo število in se ni niti enkrat zmotil glede identitete določenega govorca, če je dobil takšno

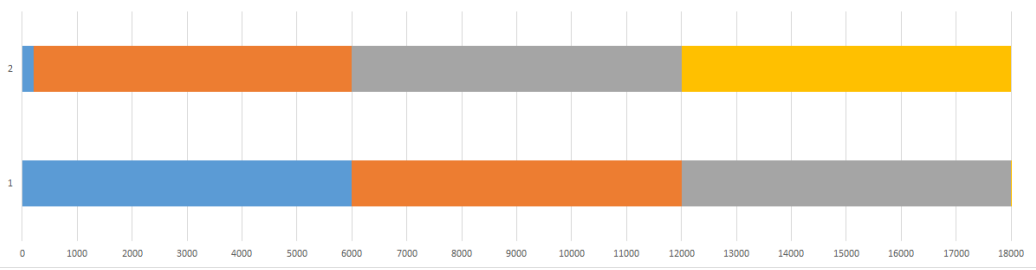
oceno.

4.4 Rezultati

V tem delu bomo pogledali rezultate vseh algoritmov, ki so bili testirani. Posnetki so bili razdeljeni na intervale $25ms$ in vsak interval je imel oznake, kateri govorec govori, ena oznaka je dala algoritem, ki se preverja, in ena je od ročne diarizacije. Če ročna diarizacija pravi, da v določenem intervalu govori več kot ena oseba in ni mogoče opredeliti, katera je ta oseba, se interval preskoči. Problem se pojavi pri zapisu govorcev. Avtomatično za vsa algoritma prvi govorec v posnetku je *govorec 1*, drugi ki začne govoriti je *govorec 2* in tako naprej, vendar če algoritem naredi napako na samem začetku in označi, da v enem segmentu na začetek govori drug govorec, potem bo zaradi poimenovanja algoritem dal zelo slabo oceno, tudi če je relativno dobro naredil svojo nalogo. Primer takšne napake pri ocenjevanju je na sliki 4.9. Zato se pri poimenovanju govorcev vzamejo vse permutacije oznak in za rezultat se izbere ta kombinacija, ki je dala najboljši rezultat.

4.4.1 LIUM - rezultati

V naslednji tabeli je prikazano dejansko število govorcev na posnetkih, in odkrite govorce.



Slika 4.9: Napaka pri fiksnem poimenovanju govorcev

ime posnetka	dejansko število govorcev	odkrite gruče	razlika
A_01.wav	2	3	+1
C_01.wav	2	2	0
D_01.wav	3	3	0
E_01.wav	5	5	0
I_01.wav	3	10	+7
J_01.wav	3	3	0
K_01.wav	3	8	+5
L_01.wav	3	6	+3

Opazimo lahko, da je LIUM zelo dobro izvedel gručenje segmentov in na polovici posnetkov odkril točno število govorcev. Večje napake so se pojavile pri *I_01.wav*, *K_01.wav* in *L_01.wav*, oziroma pri posnetkih, ki imajo govorce, ki govorijo približno enako časa in imajo prekrivanja. Seveda število gruč ne pomeni, da je algoritem dal dobre ali slabe rezultate, ker so nekatere gruče mogoče tako majhne, da so zanemarljive v primerjavi z ostalim. Točnost diarizacije je podana v naslednji tabeli:

ime posnetka	točnost
A_01.wav	81,45%
C_01.wav	82,40%
D_01.wav	46,85%
E_01.wav	73,72%
I_01.wav	44,62%
J_01.wav	77,31%
K_01.wav	49,34%
L_01.wav	54,20%

Opazimo lahko, da je algoritem dal zelo dobre rezultate za te posnetke, na katerih je točno ocenil število govorcev. Edina izjema je posnetek *D_01.wav*, na katerem pride do pogosta prekrivanja med govorcem. Seveda je dal slabše rezultate za ta posnetek, na katerem je dal napačno število govorcev z najslabšo oceno pri *I_01.wav*, kjer je dal zelo visoko oceno za število govorcev.

Če želimo ugotoviti, ali je boljše, če algoritem dela strožje gručenje, lahko združimo skupaj nekaj gruč in pogledamo, ali se rezultat izboljša:

ime posnetka	združene gruče	originalna točnost	nova točnost
A_01.wav	2	81,45%	85,36%
I_01.wav	8	44,62%	87,50%
K_01.wav	6	49,34%	86,63%
L_01.wav	4	54,20%	88,41%

Vidimo, da se rezultat izboljša za vse posnetke, kjer je to možno, tako da v takšnem primeru je dobro, da algoritem nadaljuje z gručenjem, dokler ne dobi manjšega števila govorcev.

4.4.2 PyAA - rezultati

V naslednji tabeli je prikazano dejansko število govorcev na posnetkih, in odkrite govorce.

ime posnetka	dejansko število govorcev	odkrite gruče	razlika
A_01.wav	2	2	0
C_01.wav	2	2	0
D_01.wav	3	3	0
E_01.wav	5	2	-3
I_01.wav	3	2	-1
J_01.wav	3	2	-1

K_01.wav	3	4	+1
L_01.wav	3	2	-1

Opazimo lahko, da je pyAA zelo dobro izvedel gručenje segmentov in skoraj na vse posnetke točno ocenil število govorcev. PyAA je zelo dobro ocenil število govorcev na *I_01.wav*, posnetek na katerem skoraj vsi algoritmi imajo težave oceniti točno število gruč. Seveda število gruč ne pomeni, da je algoritem dal dobre ali slabe rezultate, ker so nekatere gruče mogoče tako majhne, da so zanemarljive v primerjavi z ostalim. Točnost diarizacije je podana v naslednji tabeli:

ime posnetka	točnost
A_01.wav	58,12%
C_01.wav	62,08%
D_01.wav	39,40%
E_01.wav	46,99%
I_01.wav	70,89%
J_01.wav	49,86%
K_01.wav	54,80%
L_01.wav	52,34%

Opazimo lahko, da je algoritem dal relativno dobre rezultate. Zaradi točnega števila govorcev je dal zelo dober rezultat na *I_01.wav*, najboljša ocena od vseh algoritmov, vendar je dal slabe rezultate za posnetke kot *D_01.wav*, *E_01.wav* in *J_01.wav*, oziroma posnetke, kjer večino časa govori ena oseba, dominanten govorec.

Ta algoritem ponuja možnost, da vnaprej ve, koliko govorcev je na posnetku. Če je ta parameter podan, so rezultati naslednji:

ime posnetka	podano število govorcev	originalna točnost	nova točnost
E_01.wav	5	46,99%	35,09%
I_01.wav	3	70,89%	66,93%
J_01.wav	3	49,86%	40,75%
K_01.wav	3	54,80%	61,20%
L_01.wav	3	52,34%	40,93%

Vidimo, da se rezultat ne izboljša, tako da lahko sklepamo, da je gručenje tega algoritma dobro, in če želimo izboljšati njegovo točnost, moramo pogledati v segmentaciji. Če je število govorcev vnaprej znano, algoritem da slabše rezultate, ker predvideva, da morajo biti gruče večje, vendar gre na teh posnetkih za majhne gruče.

4.4.3 Diarize-jruby - rezultati

V naslednji tabeli je prikazano dejansko število govorcev na posnetkih, in odkrite govorce.

ime posnetka	dejansko število govorcev	odkrite gruče	razlika
A_01.wav	2	3	+1
C_01.wav	2	1	-1
D_01.wav	3	1	-2
E_01.wav	5	4	-1
I_01.wav	3	9	+6
J_01.wav	3	3	0
K_01.wav	3	8	+5
L_01.wav	3	4	+1

Opazimo lahko, da je diarize-jruby odkril pravilno število govorcev na zelo malo posnetkih in je imel podobne težave kot LIUM-a. Seveda število gruč ne pomeni, da je algoritem dal dobre ali slabe rezultate, ker so nekatere gruče mogoče tako majhne, da so zanemarljive v primerjavi z ostalimi. Točnost diarizacije je podan v naslednji tabeli:

ime posnetka	točnost
A_01.wav	81,45%
C_01.wav	72,48%
D_01.wav	88,57%
E_01.wav	80,52%
I_01.wav	59,17%
J_01.wav	77,46%
K_01.wav	49,34%
L_01.wav	84,67%

Opazimo lahko, da je algoritem dal izjemno dobre rezultate, kljub temu da ni imel najboljšega gručenja, in je uspel dati zelo dobre ocene na vseh posnetkih. Edine težave, ki jih je imel, so pri *I_01.wav* in *K_01.wav*, kot ostali algoritmi, oziroma pri posnetkih, pri kateri pride do pogoste menjave govorca in prekrivanja.

Če želimo ugotoviti, ali je boljše, če algoritem dela strožje gručenje, lahko združimo skupaj nekaj gruč in pogledamo, ali se rezultat izboljša:

ime posnetka	združene gruče	originalna točnost	nova točnost
A_01.wav	2	81,45%	85,36%
I_01.wav	7	59,17%	87,50%
K_01.wav	6	49,34%	86,25%
L_01.wav	2	84,67%	88,41%

Vidimo, da se rezultat izboljša za vse posnetke, kjer je to možno, tako da v takšnem primeru je dobro, da algoritem nadaljuje z gručanjem, dokler ne dobi manjšega števila govorcev. Lahko tudi opazimo, da so rezultati zelo podobni kot pri LIUM (samo če združujemo, ne samostojno), tako da lahko sklepamo, da velike razlike v algoritmu v tem primeru ni.

4.4.4 VoiceID - rezultati

V naslednji tabeli je prikazano dejansko število govorcev na posnetkih, in odkrite govorce.

ime posnetka	dejansko število govorcev	odkrite gruče	razlika
A_01.wav	2	5	+3
C_01.wav	2	5	+3
D_01.wav	3	5	+2
E_01.wav	5	6	+1
I_01.wav	3	9	+6
J_01.wav	3	3	0
K_01.wav	3	8	+5
L_01.wav	3	6	+3

Opazimo lahko, da je VoiceID zelo slabo ocenil število gruč z relativno veliko razliko tudi pri posnetkih, ki so razumljivi in brez prekrivanj. Vendar seveda število gruč ne pomeni, da je algoritem dal dobre ali slabe rezultate, ker so nekatere gruče mogoče tako majhne, da so zanemarljive v primerjavi z ostalimi. Točnost diarizacije je podana v naslednji tabeli:

ime posnetka	točnost

A_01.wav	48,99%
C_01.wav	68,16%
D_01.wav	36,92%
E_01.wav	60,27%
I_01.wav	38,13%
J_01.wav	89,88%
K_01.wav	51,41%
L_01.wav	45,04%

Opazimo lahko, da je algoritem dal relativno dobre rezultate. Na nekatere posnetke je dal zelo dobre rezultate, vendar je za posnetke, kot je *I_01.wav*, kjer je dal napačno število govorcev in *D_01.wav*, kjer je veliko število prekrivanj med govorci, dosegel slabo oceno.

Če želimo ugotoviti, ali je boljše, če algoritem dela strožje gručenje, lahko združimo skupaj nekaj gruč in pogledamo, ali se rezultat izboljša:

ime posnetka	združene gruče	originalna točnost	nova točnost
A_01.wav	4	48,99%	85,21%
C_01.wav	4	68,16%	80,96%
D_01.wav	3	36,92%	72,51%
E_01.wav	2	60,27%	83,92%
I_01.wav	7	38,13%	89,08%
K_01.wav	6	51,41%	87,94%
L_01.wav	4	45,04%	88,03%

Vidimo, da se rezultat bistveno izboljša za vse posnetke, tako da je dobro, če algoritem nadaljuje z gručenjem, dokler ne dobi manjšega števila govorcev.

4.4.5 Alize - rezultati

V naslednji tabeli je prikazano dejansko število govorcev na posnetkih, in odkrite govorce.

ime posnetka	dejansko število govorcev	odkrite gruče	razlika
A_01.wav	2	3	+1
C_01.wav	2	1	-1
D_01.wav	3	1	-2
E_01.wav	5	1	-4
I_01.wav	/	/	/
J_01.wav	3	1	-2
K_01.wav	3	1	-2
L_01.wav	3	1	-2

Alize je izjemno slabo ocenil število gruč. Skoraj na vseh posnetkih je ocenil, da je en sam govorec in sploh ni uspel oceniti števila govorcev pri *I_01.wav* Vendar seveda število gruč ne pomeni, da je algoritem dal dobre ali slabe rezultate, ker so nekatere gruče mogoče tako majhne, da so zanemarljive v primerjavi z ostalimi. Točnost diarizacije je podana v naslednji tabeli:

ime posnetka	točnost
A_01.wav	68,11%
C_01.wav	77,12%
D_01.wav	56,29%
E_01.wav	6,65%
I_01.wav	/
J_01.wav	1,73%
K_01.wav	42,74%
L_01.wav	68,41%

Opazimo lahko, da je algoritem dal mešane rezultate. Za nekatere posnetke, kot so *A_01.wav*, *C_01.wav*, *D_01.wav* in *L_01.wav*, je dal relativno dobre rezultate, vendar pa je dal za posnetke, kot so *E_01.wav*, *I_01.wav* in *J_01.wav*, izjemno slabe rezultate in ni mogel ugotoviti, da se sploh kdo pogovarja na posnetku, ter ni uspel najti govorcev večino časa.

4.4.6 MAAL - rezultati

ime posnetka	dejansko število govorcev	odkrite gruče	razlika
A_01.wav	2	5	+3
C_01.wav	2	5	+3
D_01.wav	3	5	+2
E_01.wav	5	6	+1
I_01.wav	3	9	+6
J_01.wav	3	3	0
K_01.wav	3	8	+5
L_01.wav	3	6	+3

Opazimo lahko, da je MAAL relativno slabo ocenil število gruč z velike razlike tudi pri posnetkih, ki so razumljivi in brez prekrivanj. Vendar seveda število gruč ne pomeni, da je algoritem dal dobre ali slabe rezultate, ker so nekatere gruče mogoče tako majhne, da so zanemarljive v primerjavi z ostalimi. Točnost diarizacije je podana v naslednji tabeli:

ime posnetka	točnost
A_01.wav	61,74%
C_01.wav	25,28%
D_01.wav	50,82%

E_01.wav	55,80%
I_01.wav	64,24%
J_01.wav	58,82%
K_01.wav	45,19%
L_01.wav	60,75%

Opazimo lahko, da je algoritem dal relativno dobre rezultate. Skoraj vsi posnetki imajo relativno dobro oceno v primerjavi z drugimi algoritmi, vendar je dosegel zelo slabo oceno na posnetku *C_01.wav* zaradi pogostosti spreminjanja govorcev in šibkejšega glasu drugega govorca, ki je verjetno predstavljal največji problem.

Ta algoritem ponuja možnost, da vnaprej ve, koliko govorcev je na posnetku. Če je ta parameter podan, so rezultati naslednji:

ime posnetka	podano število govorcev	originalna točnost	nova točnost
E_01.wav	2	25,28%	60,16%
E_01.wav	3	50,82%	38,24%
E_01.wav	5	55,80%	32,92%
I_01.wav	3	64,24%	78,48%
J_01.wav	3	58,82%	44,65%
K_01.wav	3	45,19%	59,32%
L_01.wav	3	60,75%	37,94%

Vidimo, da dobimo mešane rezultate. Za nekatere posnetke dobimo boljšo oceno, tako da bi bilo zanje dobro, da se nadaljuje z gručenjem, za nekatere pa bi morali iskati razlog v sami segmentaciji, če bi želeli izboljšati rezultat.

4.5 Primerjava algoritmov in analiza rezultatov

V tem poglavju bomo naredili primerjavo med algoritmi za vsak zvočni posnetek posebej (slika 4.10), potem pa jih bomo primerjali glede povprečne točnosti vseh posnetkov (slika 4.11).

Na prvem posnetku (*A_01.wav*) lahko opazimo vidne razlike med algoritmi, kjer dobimo več kot 30% razlike med VoiceID in algoritmoma LIUM ter diarize-jruby. Ta dva algoritma v tem primeru imata isto oceno, ker diarize-jruby uporablja LIUM za osnovo svojih algoritmov in v tem primeru očitno ne pride do bistvenih sprememb. Razlog, zakaj imajo ostali algoritmi slabšo oceno, je, da so pričakovali, da je razmerje med govorcem manjše, kot dejansko je, kar se tudi dogaja pri VoiceID, kljub temu da uporablja LIUM za osnovo.

Na drugem posnetku *C_01.wav* je največja razlika, ki jo opazimo, slaba ocena MAAL-a. Razlog za to je šibkejši glas enega govorca, kar je povzročilo težave pri njegovi detekciji s strani MAAL-a.

Na posnetku z imenom *D_01.wav* imamo zelo dobro oceno diarize-jrubya. Razlog za to so dodatne lastnosti, ki jih diarize implementira (KUllback-Leiber divergentna aproksimacija, supervektorje in podobno), s katerim je dobro izvedel segmentacijo ne glede na moč govora posameznega govorca.

Na četrtem posnetku se je prvič pojavilo presenečenje z Alize, ki se nadaljuje v naslednja dva posnetka. Razlogi za ta slab rezultat so: zelo malo število najdenih govorcev, zelo majhna dolžino segmentov v *E_01.wav* in *J_01.wav* ali čisto nasprotno - zelo velika dolžina segmentov v *K_01.wav*. Pri tem je dal skoraj celotni posnetek v eno gručo, kar pomeni, da ni zaznal sprememb govorcev.

Pri *J_01.wav* dominirata LIUM in diarize/VoiceID, ki sta na osnovi LIUM-a, kar pomeni, da ima LIUM boljši algoritem za segmentacijo in gručenje, če so govorci nesorazmerni en drugemu oziroma en govori več kot ostalih.

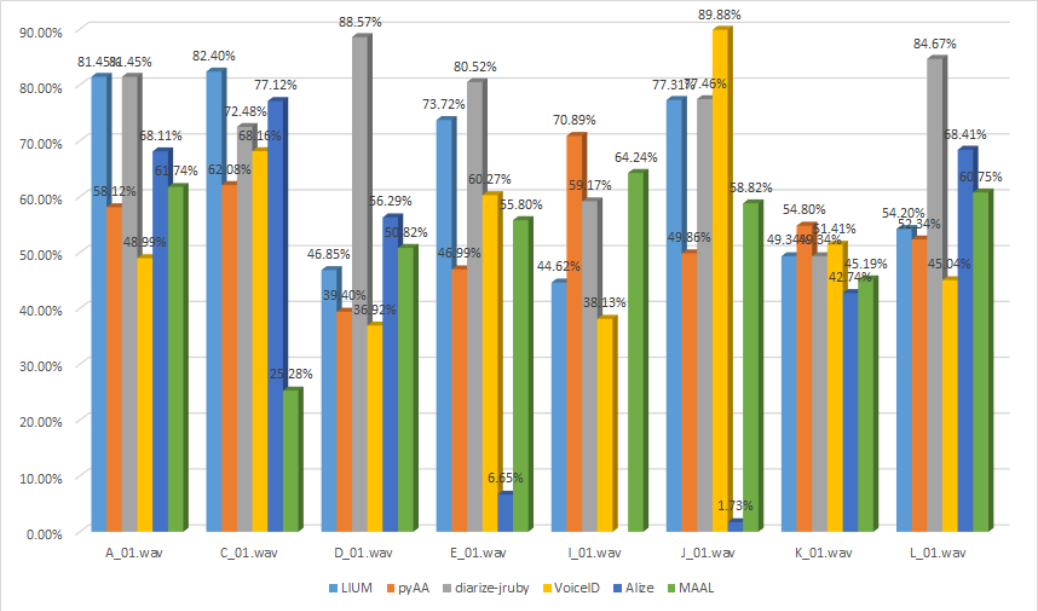
Pri *K_01.wav* noben algoritem ne izstopa preveč, razlog za to pa je ver-

jetno posnetek, ki ima veliko prekrivanj in, razen na teh časovnih intervalih, kjer je govor določenega govorca čist, ne dopušča prostora za dokazovanje.

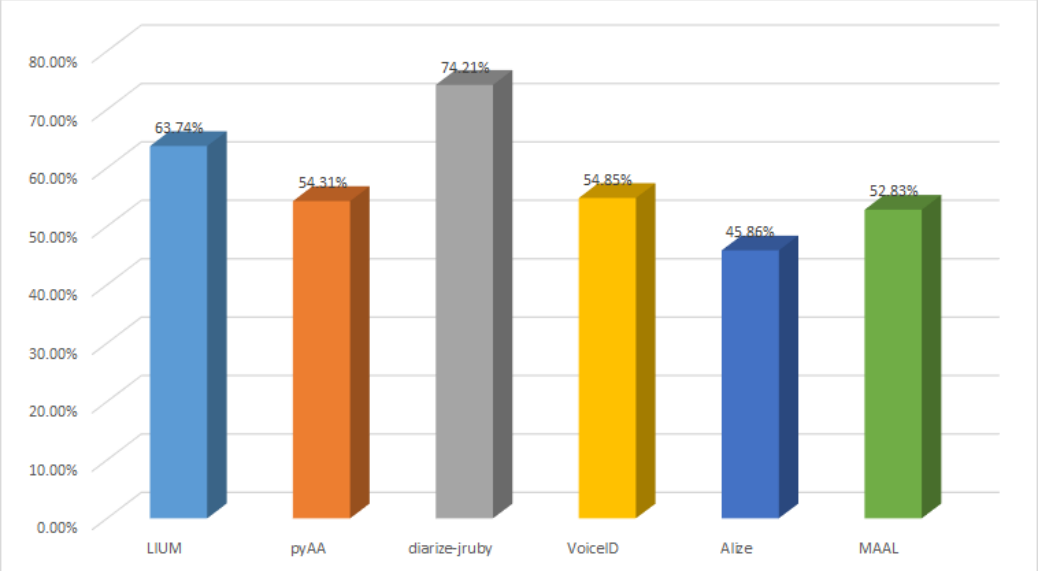
Na zadnjem posnetku pa vidimo še eno dominacijo od diarize-jruby, ki je pokazal, da je tudi zelo dober za posnetke, ki imajo nesorazmerno dolge govorne intervale med osebami na posnetku.

V naslednji tabeli so povzete prednosti algoritmov pred ostalim:

ime algoritma	prednosti pred ostalim algoritmom
LIUM	dobro dela, ko velikosti gruč niso sorazmerne, ko se govorci šibkeje slišijo.
PyAudioAnalysis	dobro dela, ko najprej veliko govori en govorec, potem drugi in tako naprej.
Diarize-jruby	dobro dela, ko velikosti gruč niso sorazmerne, ko se govorci šibkeje slišijo, ko določeni govorci govorijo bistveno močneje in ko imamo dominacijo enega govorca.
VoiceID	dobro dela, ko velikosti gruč niso sorazmerne, ko se govorci šibkeje slišijo in ko gre za film ali oddajo.
ALIZE	dobro dela, ko se govorci šibkeje slišijo, če gre za detekcijo govora ali verifikacijo (primer glasovna prijava).
Matlab Audio Analysis Library	dobro dela, ko posnetek ni dinamičen in ne pride do pogoste spremembe govorcev, ko najprej veliko govori en govorec, potem drugi in tako naprej.



Slika 4.10: Primerjava algoritmov za posamezne posnetke



Slika 4.11: Primerjava algoritmov glede povprečne točnosti

Poglavje 5

Sklepne ugotovitve

Metode za diarizacijo, oziroma ekstrakcija značilk, segmentacijo in na koncu gručenje so zelo privlačno področje v zadnjih nekaj letih. Zaradi hitrega povečanja obsega zvočnih posnetkov je segmentacija govorcev zelo aktivna tema raziskav [3]. V tej diplomski nalogi so bili pregledani in evalvirani eni izmed najbolj priljubljenih algoritmov za diarizacijo. Čeprav mnogi implementirajo zelo podobne algoritme, so imeli različen pristop in namene. Ta diplomska naloga je bila narejena z ambicijo, da služi kot izhodišče in prvi koraki na to zanimivo raziskovalno področje. State-of-the-art[1] algoritme za segmentacijo govorcev in grozdenje delajo zadovoljivo dobro v čistih pogovorih, vendar je še vedno prostor za izboljšave, recimo, ko pogovori potekajo v hrupnih okoljih, pri posnetkih sestankov, ko razmerje med govorcem ni enako ali ko so določeni govorci tišji v primerjavi z drugimi [9] [10].

Čeprav mogoče ni najbolj smiselno primerjati poprečne ocene vseh posnetkov, je vseeno ta primerjava podana na sliki 4.11. Razlog za to so posnetki, ki se razlikujejo med seboj v njihovih značilnostih, in tudi relativno majhno število posnetkov. Vsi algoritmi imajo prednosti in slabosti za določene stvari, kar je pričakovano, ker so vsi narejeni za različne namene. Vendar je bil naš cilj od samega začetka najti algoritem, ki bo deloval na različnih podatkih, ne le na enem, ki je vnaprej določen za posamezno situa-

[1] najnovejša faza v razvoju izdelka, ki vsebuje najnovejše ideje in posodobljene lastnosti

cijo (sestanek, oddaja, film), in to je razlog, zakaj so bili vsi obravnavani ter ovrednoteni na istem načinu.

Seveda je zdaj smiselni korak izdelati lasten algoritem za diarizacijo, ki bo optimiziran za slovensko jezikovno območje, in tudi, če ne uspemo dobiti boljših rezultatov, bomo razširili znanje na tem področju ter pustili vrata za nadaljnje znanje odprta.

Literatura

[1] Margarita Kotti, Vassiliki Moschou, and Constantine Kotropoulos. Speaker segmentation and clustering. *Signal processing*, 88(5):1091–1124, 2008.

[2] Marijn Anthonius Henricus Huijbregts. Segmentation, diarization and speech transcription: surprise data unraveled. 2008.

[3] Jonathan G Fiscus, Jerome Ajot, and John S Garofolo. The rich transcription 2007 meeting recognition evaluation. In *Multimodal Technologies for Perception of Humans*, pages 373–389. Springer, 2008.

[4] Xuan Zhu, Claude Barras, Sylvain Meignier, and Jean-Luc Gauvain. Combining speaker identification and bic for speaker diarization. In *INTERSPEECH*, volume 5, pages 2441–2444, 2005.

[5] Sylvain Meignier and Teva Merlin. Lium spkdiarization: an open source toolkit for diarization. In *CMU SPUD Workshop*, volume 2010, 2010.

[6] Jean-François Bonastre, Frédéric Wils, and Sylvain Meignier. Alize, a free toolkit for speaker recognition. In *ICASSP (1)*, pages 737–740, 2005.

[7] Jean-François Bonastre, Nicolas Scheffer, Driss Matrouf, Corinne Fredouille, Anthony Larcher, Alexandre Preti, Gilles Pouchoulin, Nicholas WD Evans, Benoit GB Fauve, and John SD Mason. Alize/spkdet: a state-of-the-art open source software for speaker recognition. In *Odyssey*, page 20, 2008.

[8] Theodoros Giannakopoulos and Aggelos Pikrakis. *Introduction to Audio Analysis: A MATLAB® Approach*. Academic Press, 2014.

[9] Sue E Tranter, Douglas Reynolds, et al. An overview of automatic speaker diarization systems. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5):1557–1565, 2006.

[10] Douglas A Reynolds, Patrick Kenny, and Fabio Castaldo. A study of new approaches to speaker diarization. In *Interspeech*, pages 1047–1050, 2009.

[11] Mickael Rouvier, Grégor Dupuy, Paul Gay, Elie Khoury, Teva Merlin, and Sylvain Meignier. An open-source state-of-the-art toolbox for broadcast news diarization. Technical report, Idiap, 2013.

[12] Zhu Liu, Yao Wang, and Tsuhan Chen. Audio feature extraction and analysis for scene segmentation and classification. *Journal of VLSI signal processing systems for signal, image and video technology*, 20(1-2):61–79, 1998.